ROBUST MULTICAST PROTOCOLS FOR WIRELESS MULTIHOP NETWORKS

A Dissertation
Presented to
The Academic Faculty

by

Daniel Lertpratchya

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2014

ROBUST MULTICAST PROTOCOLS FOR WIRELESS MULTIHOP NETWORKS

Approved by:

Professor Douglas M. Blough,
Committee Chair
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Douglas M. Blough, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor George F. Riley
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henry L. Owen
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raheem A. Beyah
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Geoffrey Ye Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Ellen W. Zegura
School of Computer Science
Georgia Institute of Technology

Date Approved: May 2014

To my parents, Dr. Preecha and Dr. Aporn;

and

my sister, Alisa

ACKNOWLEDGEMENTS

First, I would like to extend my gratitude to my advisor, Dr. Douglas M. Blough, and my co-advisor, Dr. George F. Riley, for their guidance, support, and patience throughout my graduate study. I also would like to thank my dissertation committee members, Dr. Henry L. Owen III, Dr. Raheem A. Beyah, Dr. Geoffrey Ye Li, and Dr. Ellen W. Zegura for their valuable advice and suggestions.

Many people have helped me throughout my graduate study – academically and non-academically, knowingly and unknowingly. I would like to thank Brian Swenson for his help with ns-3, Luis Miguel Cortés-Peña for his help with wireless networks, Safayet Ahmed for his help with Linux kernel development; and Sampan Nettayanun and Jomsurang Ruangprapun for their help with statistics. I would also like to extend my warm thanks to Sirinart Tangruamsub, Thanawadee Preeprem, Ramya Srinivasan, Nawarat Termtanasombat, Krist Wongsuphasawat, Brian Hayes, Gunn Vanichbuncha, and Natthapol Prakongpan for their help and encouragement during my study.

Last, but most importantly, I would like to extend my deepest gratitude to my parents, Dr. Preecha Lertpratchya and Dr. Aporn Lertpratchya, and my little sister, Alisa Lertpratchya, for their unconditional love, encouragement, understanding, and support throughout the years. Without my parents and my sister, this PhD definitely would not have been possible at all, and I dedicate this PhD to them.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . . **iv**

LIST OF TABLES . . . . . **ix**

LIST OF FIGURES . . . . . **x**

SUMMARY . . . . . **xii**

I INTRODUCTION **1**

1.1 Contributions 2

1.2 Dissertation organization 3

II ORIGIN AND HISTORY OF THE PROBLEM **4**

2.1 Network layer multicast 4

2.1.1 Routing structure 5

2.1.2 Creating and maintaining routing structure 11

2.2 MAC layer multicast 12

2.3 Wireless network simulations 14

III SIMULATING FRAME-LEVEL BURSTY LINKS IN WIRELESS NETWORKS **16**

3.1 Modeling bursty link behavior 17

3.2 Bursty probability adjustment function and cache TTL 19

3.2.1 Trace file-based values 19

3.2.2 Synthetic-link values 23

3.2.3 Algorithm complexity 26

3.3 ns-3 implementation 26

3.4 Evaluation 29

3.4.1 Trace files generation 29

3.4.2 Evaluation against real wireless links 30

3.4.3 Simulating links with different bursty characteristics 34

3.4.4 Effect of the simulated bursty link on routing protocols . . . 36

3.5 Discussion 41

IV LINK DISCOVERY WITH RELIABLE MULTICAST PROTOCOL 42

4.1 Link discovery protocol and reliable multicast 42

4.1.1 Basic reliable multicast protocol 43

4.1.2 Link discovery 47

4.1.3 Reliability and scalability 49

4.2 Performance evaluation 50

4.2.1 Simulation parameters and assumptions 50

4.2.2 The idealized neighborhood relationship 51

4.2.3 Evaluation metrics 52

4.2.4 Speed of link discovery and its impact 53

4.2.5 Unidirectional and bidirectional links 56

4.2.6 Effect of the application traffic model 58

4.2.7 Threshold-based transaction 59

4.2.8 Overhead of positive acknowledgement approach 60

4.3 Discussion 61

V INTERFERENCE-AWARE MULTICAST FOR WIRELESS MULTIHOP NETWORKS 63

5.1 System model and problem formulation 65

5.2 Interference-aware routing for low intensity multicast 65

5.2.1 Branching nodes with two children 66

5.2.2 Branching nodes with three children 68

5.2.3 Interference-aware Steiner tree algorithm (IAST) 69

5.3 Interference-aware routing for general multicast 71

5.3.1 Scaling factor in path nodes 71

5.3.2 Fixed-distance tree merging algorithm 73

5.3.3 Scheduling algorithm 74

5.4 Performance evaluation 74

5.4.1 Simulation parameters and assumptions 74
5.4.2 Low intensity multicast 75
5.4.3 General multicast 76
5.4.4 Contention-based channel access 79
5.4.5 Bursty wireless channels 80
5.5 Discussion 81

VI INTERFERENCE-AWARE MESH MULTICAST FOR WIRELESS MULTIHOP NETWORKS 82

6.1 Overlay link extension algorithm 83
6.1.1 Mesh branching nodes 84
6.1.2 Mesh merging nodes 85
6.2 Delaunay mesh extension algorithm 87
6.2.1 Delaunay mesh extension with nodes merging 88
6.3 Performance evaluation 89
6.3.1 Simulation parameters and assumptions 89
6.3.2 Tuning scaling factor and merging distance 91
6.3.3 Performance of multicast routing structures with bursty links 93
6.3.4 Link burstiness and its impact 95
6.3.5 Performance of multicast routing structures with faulty nodes 96
6.4 Discussion 98

VII CONCLUSION AND FUTURE WORK 99

7.1 Contributions 99
7.2 Future work 100
7.2.1 Simulating frame-level bursty links in wireless networks . . . 100
7.2.2 MAC layer reliable multicast and link discovery 101
7.2.3 Interference-aware multicast 101

APPENDICES 103

APPENDIX A — TRACE FILES INFORMATION 104

APPENDIX B — BURSTY LINK SIMULATION RESULTS . . . 109

APPENDIX C — PUBLICATIONS 114

REFERENCES 115

LIST OF TABLES

1 Simulation parameters used to evaluate MAC layer multicast protocols. 51

2 Simulation parameters used to evaluate multicast routing structures. 75

3 Simulation parameters used to evaluate tree extension structures. . . 91

4 Trace files information. 104

5 Burstiness metrics of the 100 trace files. 109

LIST OF FIGURES

1 An example of CPDF of a link with bursty behavior. 20
2 An example of calculating bursty probability adjustment from CPDF. 21
3 Examples of possible synthetic bursty probability adjustment functions. 23
4 Examples of different scaled-erf functions. 24
5 Complementary cumulative distribution function of $|\beta|$ values. 31
6 A comparison between simulated links and trace files. 32
7 Packet reception ratio for the trace file and different models. 32
8 CPDF from one trace file along and its simulated links. 33
9 Distribution of burst sizes from the trace file and different models. . . 33
10 Average packet reception ratio and received power trace. 34
11 Using a scaled-erf function to simulate links with different burstiness. 35
12 A diamond topology used to study performances of different models. . 37
13 Packet reception ratios of different models with varying link quality. . 38
14 DSR simulation results with different bursty link models (UDP). . . . 40
15 DSR simulation results with different bursty link models (TCP). . . . 40
16 Modified 802.11 headers used by LDM protocol. 44
17 Two types of transaction of LDM protocol. 45
18 Add delay for each MAC layer multicast protocol under different SNR. 53
19 Average add delay of each protocol. 54
20 Multicast packet reception ratio for each protocol. 55
21 Add delay of LDM with different maximum Join ACK probabilities. . 55
22 Total number of transmissions resulting from misclassification. 58
23 Average throughput for different MAC layer multicast protocols. . . . 59
24 Transmission time used for different frame types. 60
25 Synchronization of frames forwarding after a multicast tree branch. . 62
26 A branching node u branching into two paths. 66
27 Values of b_2 with different branching angle θ. 67

28 A branching node s branching into three paths. 68
29 An example of building a multicast tree by IAST algorithm. 70
30 An infinitely-long, equally-spaced linear network. 72
31 End-to-end delivery delay of different multicast routing structures. . . 75
32 IAST schedule length with different scaling factors. 76
33 Schedule length with varying number of destinations. 77
34 Schedule length of different algorithms with varying node density. . . 78
35 CSMA goodput of different multicast routing structures. 79
36 MPRR of the IAST protocol with varying wireless link burstiness. . . 80
37 The underlying idea of the overlay link extension. 84
38 Two scenarios to be considered of the overlay link extension algorithm. 84
39 Values of m with different $\mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}$ and α. 85
40 An example of OLE algorithm. 86
41 An example of DME algorithm. 88
42 Examples of different interference-aware multicast routing structures. 90
43 MPRR of OLE and DME with different scaling factors. 92
44 MPRR of DME-merge algorithm with different merging distance. . . 92
45 MPRR of different mesh multicast structures. 94
46 Number of links added to the multicast tree from different algorithms. 95
47 MPRR of multicast routing structures with different burstiness. . . . 97
48 MPRR of different algorithms at varying faulty node probability. . . . 98

SUMMARY

Wireless ad hoc networks are self-organizing networks that do not rely on any infrastructure to operate. One of the common communication schemes that is particularly important is multicasting where a single node sends a message to multiple destinations. Instead of sending multiple copies to each destination individually, multicasting allows the message to be sent to all intended destinations in one operation.

Multiple multicast routing protocols have already been proposed. However, most of the goals of multicast routing protocols are to create and maintain the multicast routing structures efficiently. Even though interference is an important characteristic of wireless communication, most of the currently available multicast routing protocols do not take interference into account when building the multicast routing structures.

The objective of this dissertation is to study the problem of multicasting in wireless ad hoc networks, while taking the interference from other nodes into account. We classify nodes in the multicast routing structures into different classes based on their role in the multicast routing structures. We derive the optimal routing strategies for different classes, using the most accurate interference model. Based on the analyses, we propose four interference-aware multicast routing algorithms. We evaluate the performance of our proposed algorithms using wireless network simulation, where we have implemented an accurate model of wireless communication. We show that, by taking interference into account when building the multicast routing structures, our proposed multicast routing algorithms are able to improve the performance of multicasting over other multicast routing algorithms that do not consider wireless interference when building multicast routing structures.

CHAPTER I

INTRODUCTION

Wireless ad hoc networks are self-organizing networks that do not rely on any infrastructure to operate. In these networks, two nodes can communicate directly if they are within the transmission range of each other. Otherwise, they rely on other nodes in the network to route their packets. One of the routing services that is particularly important is multicast routing where packets are to be delivered to multiple intended destinations. Multicast routing service plays a crucial role in various applications such as data distribution, video conferencing, and communications in military operations. Instead of sending data via multiple unicasts, multicasting minimizes channel capacity consumption by sending data to intended destinations in a single operation. The use of multicasting can reduce the cost of communication and improve efficiency in the network.

Multicasting in wireless network ad hoc networks is more complex than traditional wired networks and presents several challenges. By nature, wireless communication is more vulnerable to noise and interference from transmissions from other nodes than wired communication. As a result, multicast protocols designed for wired-networks are not suitable for wireless networks. Recently, several multicast routing protocols designed for wireless networks have been proposed. While these protocols provide better performance as compared to using multiple unicasts, their main goal is to route packets from a source to destinations efficiently, or to reduce overhead of control messages in the network.

1.1 Contributions

We propose to design robust multicast protocols. Unlike previous work (with a few exceptions) where reduction of interference was implicitly achieved by reducing a node's transmission power or creating a sparse topology, we explicitly take interference into account in designing the protocol. The multicast routing algorithm will employ a hierarchical routing approach with the goal to reduce interference experienced by the receivers in the network. Reducing interference in the network can increase overall network capacity by permitting more spatial reuse and reducing the number of packet retransmissions due to collisions. The primary contributions of this work are:

- To improve accuracy of wireless network simulations, we propose a stochastic bursty-link model to simulate bursty behavior observed in real wireless communication. The underlying idea of our stochastic bursty-link model is that the probability of successfully receiving a frame is dependent on the history of the previous receptions. We show that our model can closely simulate real wireless links with different bursty behaviors.

- We proposed an extension for IEEE 802.11 MAC layer that provides reliability for multicast transactions and incorporates a neighborhood maintenance mechanism. Our proposed extension uses positive acknowledgement mechanism to provide reliability to multicast transaction at the MAC layer. Our proposed extension also provides a mechanism for devices to quickly recognize changes in their neighbor sets, which simplify the design of higher layers and eliminate potential redundancies in their execution.

- We consider the problem of interference-aware multicast routing tree in wireless multihop networks where we classify nodes into different classes and derive optimal interference-aware routing strategies for each class. Based on the analyses, we propose new interference-aware multicast routing structures. We evaluate

our proposed structures through simulation in both the TDMA and CSMA/CA settings. Simulation results demonstrate that our proposed interference-aware multicast trees can reduce the schedule length in TDMA networks and increase goodput in CSMA/CA networks.

- We propose two algorithms to extend the interference-aware multicast tree to form an interference-aware multicast mesh. We evaluate both interference-aware mesh structures where we design the simulation with the goal to evaluate the performance of different multicast routing structures in two possible scenarios of graph disconnection – link failure and node failure. We show that our proposed interference-aware mesh structures outperform the interference-aware multicast tree and other mesh structures that do not take interference into consideration.

1.2 Dissertation organization

The remainder of this dissertation is organized as follows. Chapter 2 provides an overview of the background and related work in the areas of wireless multicasting and wireless network simulation. Chapter 3 gives a description of our wireless link model to improve the accuracy of simulating wireless links by incorporating bursty behavior observed in real wireless communications. In Chapter 4, we propose an extension to IEEE 802.11 to support reliable multicasting and neighborhood maintenance at the MAC layer. In Chapter 5, we consider network-wide multicasting by analyzing the optimal routing strategies for multicast trees. We propose two algorithms to build a multicast tree based on the analyses. We built on the multicast tree to form a multicast mesh in Chapter 6, where we analyze the optimal routing strategies for mesh structures. We propose two algorithms to build a multicast mesh from the multicast tree. Finally, Chapter 7 concludes this dissertation.

CHAPTER II

ORIGIN AND HISTORY OF THE PROBLEM

The proposed research addresses issues within the field of wireless multicasting. Two major areas directly related to the proposed research are routing and wireless network simulation. Routing concerns moving packets from a source node to a set of destination nodes. With the scale and complexity of the wireless ad-hoc networks and sensor networks, network simulation has become an indispensable tool in studying wireless network performances. In this chapter, we briefly review the concepts of routing and wireless network simulation.

2.1 Network layer multicast

Routing is a process of selecting paths to send packets from a source to a destination, possibly through other intermediate nodes. Routing can be classified as unicast routing, multicast routing, and broadcast routing. Unicast routing delivers packets to a single destination while multicast routing delivers packets to a subset of nodes within the network. Broadcast delivers packets to all nodes in the network.

Unlike traditional wired-networks, wireless ad hoc networks present different design challenges since they do not rely on any fixed infrastructure. Nodes in wireless ad hoc networks have to coordinate together to accomplish the desired goals. The coordination between nodes is even more challenging since wireless communication is not perfect; thus, routing paths can be affected by the change in wireless channel conditions. As a result, protocols designed for wired networks are not suitable for wireless ad hoc networks.

Various multicast routing protocols designed specifically for wireless ad hoc networks have been proposed. A survey of multicast protocols for ad hoc networks can

be found in [45]. Multicast routing protocols can be classified according to a wide variety of characteristics such as initialization approach, the routing structure used to route the packets, and the method of constructing and maintaining the routing structure.

2.1.1 Routing structure

Multicast routing protocols can be classified into four main categories according to their routing structures: tree-based protocols, mesh-based protocols, stateless protocols, and hybrid protocols. Tree-based protocols [43, 70, 76, 96] use different kinds of trees as underlying routing structure to route multicast messages to all destinations. Tree structures provide simple and cost effective routing infrastructures at the cost of robustness in the presence of node and link failures. Mesh-based protocols [25, 52, 88] use mesh structures to provide robustness by having multiple routes between the source and destinations at the cost of mesh structure maintenance. Structure-less multicast protocols do not explicitly create a routing structure but rely on other methods such as network coding [58, 68] and geographic routing [48, 78].

2.1.1.1 Tree-based protocols

In tree-based protocols, only one route from a source to each destination exists. Tree-based protocols can be further classified as source-tree protocols and shared-tree protocols. Source-tree protocols construct a different multicast tree for each source, whereas multiple sources share the same multicast tree in shared-tree protocols. Source-tree protocols have an advantage in that each source gets its own multicast tree that can be tailored to match the source's requirements. This advantage comes at the cost of maintaining multiple trees in the network. Shared-tree protocols construct only one tree rooted at a special node called a rendezvous point (RP) and let all sources share the same multicast tree. Shared-tree protocols are usually more

scalable than source-tree protocols since only one shared-tree is needed to be maintained. However, the single shared tree can become a bottleneck within the network. All tree-based protocols are vulnerable to varying wireless channel conditions and node mobility, as node movement might break the link between two nodes, resulting in a disconnected network. Several multicast routing protocols can be classified as tree-based protocols. Examples of tree-based protocols are [19,20,42,67,76,87,93,96].

One example of a shared-tree protocol is multicast ad hoc on-demand distance vector (MAODV) protocol [76], which is an extension of AODV unicast routing protocol [71]. In MAODV, each multicast group has one shared-tree and a group leader. The group leader's responsibility is to maintain the multicast tree by using a sequence number to ensure the freshness of routes and prevent routing loops. If a node wants to join the multicast group, it sends out a route request (RREQ) message to the group leader if the node knows the address of the group leader. Otherwise, it broadcasts RREQ throughout the network. Only the group leader or nodes already participating in the multicast group are allowed to respond to RREQ by sending a route reply (RREP) message back to the joining node. The joining node selects the shortest path among all RREPs received and sends a multicast activation (MACT) message back to the selected RREP sender. When a link breakage is detected, the tree becomes disconnected, and the downstream node is responsible for finding an alternate path to the multicast tree. The downstream node broadcasts RREQ again to repair the disconnected tree. Only the participating nodes with a lower hop count to the group leader are allowed to reply with RREP. This requirement is introduced to prevent a routing loop. If the downstream node does not receive any RREP, it assumes that the network has been partitioned and makes itself a multicast group leader.

Adaptive demand-driven multicast routing (ADMR) protocol [41] is an example of a source-tree protocol. Each multicast tree is maintained by a periodic KEEP-ALIVE message, which is broadcast by the source of the tree. A receiver interested

in joining a group starts by broadcasting a MULTICAST SOLICITATION message to the network. When the source receives the message, it replies by sending a unicast KEEP-ALIVE message back to the receiver. The receiver then sends a RECEIVER JOIN message back to the sender to confirm the joining. The sender also periodically floods the network with a RECEIVER DISCOVERY message to announce its presence to any interested receiver, which in turn sends back a RECEIVER JOIN message. ADMR can switch to flooding once it determines that node mobility is too high to efficiently maintain a routing structure.

2.1.1.2 Mesh-based protocols

In mesh-based protocols, a multicast mesh connecting a source to all destinations is constructed. Multiple routes are available between the source and destinations. As a result, mesh-based protocols are more robust than tree-based protocols since multiple routes are available for packet routing. However, this robustness comes at the expense of maintaining a mesh structure in the network. Mesh-based protocols can be classified as source-initiated protocols or destination-initiated protocols. Examples of mesh-based protocols include [21, 52, 66, 69, 88].

On-demand multicast routing protocol (ODMRP) [52] is a mesh-based multicast protocol that dynamically builds and maintains multicast group membership on demand. A multicast source periodically broadcasts a JOIN-REQUEST message throughout the network. When a node receives a non-duplicate JOIN-REQUEST, it updates the parent node ID and rebroadcasts the JOIN-REQUEST. If the receiver receives a JOIN-REQUEST, it creates an entry for the source in its member table and broadcasts the JOIN TABLE. When a node receives a JOIN TABLE, it checks whether its address is present in the JOIN TABLE or not. If the JOIN TABLE contains the node's address, it knows that it is on the path between the multicast source and the destinations, and is a part of a mesh. It then sets the forwarding

group flag (FG_flag) on and rebroadcasts the JOIN TABLE. The process continues until the JOIN TABLE reaches the multicast source, and a mesh is constructed. The multicast source refreshes the constructed mesh by sending a JOIN DATA message periodically. ODMRP adopts a soft state approach, which means that explicit control message is not required to leave a multicast group.

Protocol for unified multicasting through announcement (PUMA) [88] creates a shared mesh for each multicast group and selects one of the receivers as the core of the group. Each receiver connects to the core node along all available shortest paths. All nodes along shortest paths between any receiver and the core node form the mesh of the network. To send a multicast packet, the sender simply sends the packet to any of the nodes in the mesh via the shortest path. Once the packet has reached the mesh member, it is flooded within the mesh. Unlike ODMRP where every sender floods the network with control packets, only the core node floods the network. The other difference between ODMRP and PUMA is that ODMRP is sender-initiated whereas PUMA is receiver-initiated.

2.1.1.3 Stateless protocols

Both tree-based protocols and mesh-based protocols need to create and maintain the multicast routing structures used for delivering packets. This overhead can be substantial, especially in a high mobility scenario. Stateless protocols are designed to reduce the overhead of creating and maintaining a routing structure. In stateless protocols, little or no routing topology information is maintained at forwarding nodes. A stateless protocol either relies on geographical information to forward a packet or explicitly includes the intended destination list in a packet.

With the availability of the global positioning system (GPS), geographic routing has been made possible. In geographic routing, nodes use geographic information to route packets from a source to a destination. Only local information such as a

node's position and its neighbors' positions are required for geographic routing to operate. Thus, geographic routing protocols are highly scalable. On the other hand, pure geographic routing protocols can be described as greedy forwarding algorithm and as a result, can get stuck if a void is present in the network, requiring a separate mechanism to handle such a scenario. Geographic routing protocols also need to know the position of the destination to operate. This information must be provided by an external service that may not be available in all scenarios. Examples of geographic multicast routing protocols are [28, 78].

In geographic multicast routing (GMR) protocol [78], each node selects a subset of its neighbors as relay nodes to forward packets. The metric used for selecting the neighbors is the *cost over progress ratio*. The cost is equal to the number of selected neighbors, while progress is the reduction of the remaining distances to the destinations. A node running GMR needs only its own position and its neighbors' positions to select the best neighbors to forward packets to. In the case where no positive progress can be made, the node uses the face routing technique [49] to avoid the local hole in the network until positive progress is found.

Receiver-based multicast (RBMulticast) protocol [28] forwards packets through virtual nodes toward the destinations. Before forwarding a packet, each node creates multicast regions around itself. RBMulticast is not restricted to specific region creation schemes. RBMulticast determines which regions the packet should be forwarded to, and sets the destination of the packet to the virtual node of the corresponding regions. The virtual node's location can be determined in several ways, such as the geometric mean of the locations of all neighbors in that region. The node closest to the virtual node's location takes the responsibility to forward the packet. Like GMR, RBMulticast also avoids holes in the network by using a special procedure.

Differential destination multicast (DDM) [43] protocol is a multicast routing protocol that can operate in a stateless mode. In DDM, a multicast sender encodes a

list of the multicast receivers in a special DDM header on every packet. Each node then decides how to reach each destination by consulting its unicast routing table. Once the next hop nodes for forwarding the packet are found, the node reconstructs a DDM header for each next hop node and forwards the packet according to the unicast route. DDM can also work in a soft-state mode where the destinations are cached at each node; therefore, the upstream node does not have to include the list of the destinations in all packets. However, the destination lists at both the upstream node and downstream node have to be synchronized for DDM to work correctly. This synchronization is done by having the upstream node send control packets to notify the downstream node of the change in the destination list.

2.1.1.4 Hybrid protocols

Hybrid protocols are designed to combine both the advantages and disadvantages of all three approaches. There are multiple options for combining multiple approaches to create a hybrid protocol. MCEDAR [82], CAMP [30], and AMRoute [95] are examples of tree-mesh hybrid protocols, while EGMP [94], HRPM [23], and HGMR [48] are examples of protocols that combine a geographic approach and tree-based approach.

Multicast core-extraction distributed ad hoc routing (MCEDAR) protocol [82] and ad hoc multicast routing (AMRoute) protocol [95] are two examples of protocols that create and maintain a mesh topology but operate on a multicast tree on top of the created mesh. Their goals are to support the robustness of the mesh-based approach and to provide efficiency of the tree-based approach. In both protocols, a mesh infrastructure is created to support tree-based routing. MCEDAR implicitly creates a source-based multicast tree by using the core broadcast mechanism. In AMRoute, a core node is responsible for initiating a tree creation process, which can be viewed as identifying the subset of links that belong to the multicast tree.

Hierarchical rendezvous point multicast (HRPM) protocol [23] is a hybrid protocol that combines geographic routing and tree-based routing. In HRPM, the deployment area is divided into multiple grids. Each grid has one access point (AP) that acts as a leader of the grid. Every access point is in turn managed by a rendezvous point (RP) that acts as a leader for the whole network. To avoid having to keep track of the actual AP and RP, HRPM introduced the notation of *geographic hashing* that takes a multicast group ID as an input and outputs a location contained in the region. The node with the nearest position to the hashed location takes the role of AP or RP for the multicast group. A multicast receiver interested in joining the multicast group sends a JOIN message to the RP by using any geographic routing protocol. When a multicast source has packets to send, it sends an OPEN SESSION message to the RP to ask for the membership group information. The RP then sends back the list of grids that have active multicast receivers. The multicast source sends the multicast packets by using a geographic routing protocol to all APs within grids that have active multicast receivers. Once the packet has reached the intended AP, the AP creates an overlay tree within that grid and uses the tree to deliver the multicast packets to the multicast receivers.

2.1.2 Creating and maintaining routing structure

Two approaches for creating and maintaining a routing structure have been used: proactive and reactive. Proactive routing protocols actively create and maintain a routing structure in the network while reactive protocols create it when needed. Proactive protocols construct a routing structure even though no actual data transmission is ongoing. This proactive approach has an advantage in that routes are readily available for use when a source has a packet to send. Any change in the network such as the availability of a new route can be detected quickly. However, actively creating and maintaining a routing structure incurs high control overhead

for the network. As a result, only a few protocols are purely proactive. Examples of proactive protocols are CAMP [30], AMRoute [95], and M-LANMAR [97].

Reactive protocols, on the other hand, only construct a routing structure when the source has a packet to send. As a result, they avoid the high overhead of creating and maintaining the routing structure when it is not being used. Some of the drawbacks of the reactive approach are poor route adaptation and a high degree of delay in the initial route creation process. Multiple protocols can be classified as reactive protocols. Examples of reactive protocols are MAODV [76], ADMR [41], ODMRP [52], PUMA [88], and DDM [43].

2.2 MAC layer multicast

Multicast routing concerns moving a packet from a source to destinations. Moving a packet from a source to destinations is usually done by transmitting it along a path of directly connected nodes. This transmission between directly connected nodes is a function of a medium access control (MAC) layer. Thus, a multicast routing protocol's efficiency depends on the underlying MAC layer protocol. One of the functions that is particularly important in wireless networks is link layer reliability. A multicast routing protocol operating on top of a reliable MAC layer will have better performance than operating on top of an unreliable MAC layer.

A link layer reliable service is usually achieved with acknowledgments and retransmissions. For instance, the 802.11 MAC specification requires that a unicast frame be positively acknowledged by the receiver and that the transmitter retransmit the frame if an acknowledgement is not received. However, broadcast and multicast frames are not protected by such a mechanism. Thus, broadcast and multicast operations are unreliable in 802.11. To make protocols that operate above 802.11 reliable, two options are available: incorporating a reliability mechanism at upper layers and using a unicast transmission instead of a broadcast or multicast transmission. RMA [33]

is an example of a protocol that handles packet loss at the upper layer. Handling retransmission at the upper layer incurs higher delay and adds complexity to the protocol. An upper layer protocol can use multiple unicast transmissions instead of a single broadcast or multicast transaction at the MAC layer. However, using multiple unicast transmissions does not capture the benefit of a wireless channel, where multiple receivers can receive a frame with a single transmission.

Recently, various reliable multicast MAC protocols have been proposed to provide reliability for broadcast and multicast frames. All protocols can be broadly categorized into two approaches. One approach is using out-of-band signaling to provide feedback to a multicast sender. Examples of protocols employing this technique are RMAC [81], 802.11MX [37], and BPBT [22]. Out-of-band signaling can be used to provide acknowledgment, or to simply capture the channel. In RMAC [81], nodes use two busy tones. One busy tone is used by a receiver to signal other nodes that it is busy receiving another frame. The second busy tone is used to positively acknowledge a received frame. 802.11MX [37] also uses two busy tones. One busy tone is used for letting other nodes know that it is not ready to receive a frame and the other is used to send a negative acknowledge back to the sender. BPBT [22] uses a busy tone to capture the channel before going through a multicast transaction. To use out-of-band signaling, special hardware is required at each node, which may not be practical.

The second approach uses positive or negative acknowledgments without requiring any extra hardware. Several protocols employing this approach have been proposed [11, 14, 34, 40, 85, 86]. In BMW [85], a sender selects one of its neighbor nodes in a round robin fashion and goes through a four-way transaction (RTS-CTS-DATA-ACK). The selected receiver has a chance to ask for a retransmission if it missed any frames previously. BMW tries to exploit the nature of the wireless medium in that it is likely that neighbor nodes will also receive the multicast frame. Sheu et al. [80] proposed a protocol that divides the DIFS period after a multicast transmission into

slots for each receiver to randomly select for sending an acknowledgment. In BMMM and LAMM [86], a sender sends an RTS to each receiver, which replies with a CTS. The sender proceeds to send a DATA frame after all receivers have replied with a CTS. Finally, the sender sends a request for acknowledgment (RAK) to each receiver, instructing the receiver to reply with an ACK.

MMP [34] modifies a DATA frame to include all receivers' addresses. Each receiver then replies with an ACK sequentially as determined by its position in the DATA frame. If retransmission is required, the sender goes through RTS-CTS-DATA-ACK transaction to recover the loss. In MMP, CTS and ACK frames are the standard IEEE 802.11 CTS and ACK frames, which do not include the address of the receiver. The sender infers the receiver's address by the timing of CTS or ACK frames. Similarly, in multicast with bound (MWB) [40], a sender uses the RTS-CTS-DATA-ACK transaction to provide reliability. However, MWB splits receivers into groups of four receivers in each sub transaction. This limitation of four receivers was introduced to preserve the original 802.11 DATA frame format, which already has four address fields. CTS and ACK frames in MWB also include the receiver index, which is the position of the receiver in RTS or DATA frames.

2.3 Wireless network simulations

Network simulations have been used extensively to evaluate the performance of wireless networks and protocols or applications that operate on wireless networks. With the scale and complexity of the wireless ad-hoc networks and sensor networks, network simulation has become an indispensable tool in studying wireless network performances. Several network simulators are available for use in the research community and the industry. Examples of well known network simulators are ns-2 [4], ns-3 [5], OPNET [18], GTNetS [74], and GloMoSim [99]. Most, if not all, network simulators support wireless network simulation such as IEEE 802.11 to some extent.

The ability to accurately simulate wireless network is crucial to the credibility of any study that relies on network simulation. However, one of the characteristics of network simulation is that it requires some level of abstraction. Selecting the right level of detail is not a simple task. Too much detail requires time to implement, verify, and maintain the simulator. Moreover, simulation running time is usually longer with a more detailed model. A model with too little detail or too simplistic often leads to incorrect or unrealistic results [38].

One of the important components of wireless network simulation is the signal propagation model. Signal propagation loss models are used to calculate the received signal strength at a receiving device and the interference to other devices. The abstraction level of the signal propagation model can range from very simple models (e.g. random, fixed), theoretical-based models (e.g. Frii free-space [29], log-distance [26]), to very complex models (e.g. building propagation model). Most of the network simulators support theoretical signal propagation models such as Friis free-space model and the log-distance model. Stoffers and Riley showed that different propagation loss models in ns-3 yield different simulation results [84]. Kotz et al. comprehensively studied six assumptions about wireless communication that are commonly found in wireless network simulation and showed that these assumptions led to differences between simulation results and real world measurements [47]. Thus, the ability to accurately model the propagation loss in wireless network simulation is crucial to any wireless network study that relies on wireless network simulation.

CHAPTER III

SIMULATING FRAME-LEVEL BURSTY LINKS IN WIRELESS NETWORKS

One of the characteristics of wireless communication is that the wireless links are bursty. Multiple studies have shown that wireless links are bursty and that the burstiness affects the experimental results [10, 15, 44, 65]. A few studies have been done with the goal of quantifying the wireless link burstiness. At the physical layer, coherence time is the time during which the radio signal is considered to be stable. A Markov chain describing the burstiness at bit level was proposed by Mushkin and Bar-David [63]. One of the first metrics to measure link burstiness was proposed by Srinivasan et al. [83]. The authors defined a burstiness metric, β, by using conditional probability delivery function (CPDF), which can be obtained from packet delivery traces. The burstiness metric measures if a link is closer to an independent link or an ideal bursty link. The authors showed that most wireless links are bursty and β can be used to predict network protocol performance on a bursty link.

Even though link burstiness is a widely recognized characteristic of wireless communications, most of the currently available network simulators do not provide support for modeling wireless link burstiness. A few models have been proposed to model burstiness [32, 51, 92]. Lee et al. [51] proposed a noise model such that the level of noise depends on the history of the previous noises. Gómez et al. [32] proposed an auto-regression model to replicate a bursty behavior by modeling received powers where the model is obtained from analyzing trace files. A major limitation of modeling signal or noise variation is that obtaining accurate empirical data is very difficult or impractical. Vlavianos et al. [89] showed that getting an accurate measurement

of received signal strength indicator (RSSI) and signal-to-interference-plus-noise ratio (SINR) is difficult due to many factors. For example, according to 802.11 specifications, RSSI is only measured during the PLCP preamble and not the whole frame. Moreover, RSSI resolution is dependent on the device chipset [57] and often reported as an integer only. SINR, which must be derived from RSSI, inherits all inaccuracies from RSSI [89].

In this chapter, we propose a stochastic bursty-link model to simulate bursty behavior of the link at the *frame* level. By modeling link burstiness at the frame level, we are able to create bursty links and avoid the low-level inaccuracies and complexities. The underlying idea of our stochastic bursty-link model is that the probability of successfully receiving a frame is dependent on the history of the previous receptions. The model adjusts the probability of successfully receiving a frame based on the results of previous frame receptions. Our model can directly simulate a bursty behavior of a real wireless link given that an appropriate trace file is available to the model. Our model can also simulate bursty behavior by using appropriate functions if a trace file is not available. We show, through simulation, that our model can closely simulate real wireless links with different bursty behaviors.

3.1 Modeling bursty link behavior

In this section, we present the stochastic bursty-link model for wireless simulation. Our model is motivated by the observation that, in a bursty link, the probability of correctly receiving a frame depends on the frame reception history of the link. Our goal is to mimic the same behavior where the probability of successfully receiving a frame is also dependent on the history of the link.

The high level idea of our stochastic bursty-link model is as follows: the bursty-link model keeps track of transmission results as observed by a receiver on each link in an internal cache. When calculating the probability that the current transmission

will be successful, the model looks at the history of the previous transmissions and adjusts the probability according to the history. The model imposes a time-to-live (TTL) limit for a cache entry to prevent very old transmission results from affecting the probability value. If the cache is empty, the model simply uses the probability according to a default metric such as the distance between the sender and the receiver.

The adjustment to the probability values is made by consulting a function called a bursty probability adjustment function (BPA(n)) since it changes the probability values if the bursty behavior is taken into account. The parameter n is the size of the burst where $n > 0$ represents the number of consecutive received frames and $n < 0$ represents the number of consecutive missed frames. In other words, BPA(n) is the change in probability of successfully receiving the next frame given that the previous n consecutive frames were received (when $n > 0$) or missed (when $n < 0$). Note that with this definition, the burst size 0 is undefined. However, we can use BPA(0) = 0 to represent the case where the cache is empty and no adjustment is made to the probability value.

To summarize, the main ideas of our model are as follows.

1. The model keeps track of transmission results on each wireless link in an internal cache.

2. The probability of successfully receiving a frame is adjusted based on the previous transmission results.

3. Time-to-live is introduced to limit the time during which a previous transmission result can affect future transmissions.

Two major components of our stochastic bursty-link model are the bursty probability adjustment function and the time-to-live of a cache entry. Since the choice of the two parameters will have significant impact on the behavior of our model, it is

important that they are picked carefully when modeling a bursty link. In the next section, we discuss two methods to obtain appropriate values for these parameters.

3.2 Bursty probability adjustment function and cache TTL

In this section, we present two methods to obtain appropriate bursty probability adjustment function and time-to-live for a cache entry: from a trace file and from a set of predefined functions. A trace file approach is suitable when the goal is to simulate a specific link and it is possible to get a trace file from that link. We present a set of simple functions that can be used to simulate bursty link behavior in the case where getting a trace file is not possible or the goal of the simulation is simply to simulate bursty links that are not modeled after specific links.

3.2.1 Trace file-based values

The first method of obtaining a bursty probability adjustment function and cache TTL is through the analysis of a real trace file. By analyzing a real trace file, we can observe the bursty characteristic of the wireless link and obtain appropriate values for our model.

Since we are interested in analyzing burstiness of a link, we need to be able to keep track of the number of consecutive frames received or missed. One way is to keep track of the sequence number in the 802.11 header. We have to make sure that all frames of interest are sent with the same physical layer parameters; for example, wireless channel, modulation scheme, and data rate. This requirement can be met by fixing the transmission rate to a single value or by checking the Radiotap header of each frame.

Given a trace file, it is possible to calculate the conditional probability delivery function (CPDF(n)) from the trace file. CPDF(n) gives the probability of correctly receiving a frame for different burst sizes n. A positive burst size represents a number of consecutive successful frame receptions while a negative burst size represents a

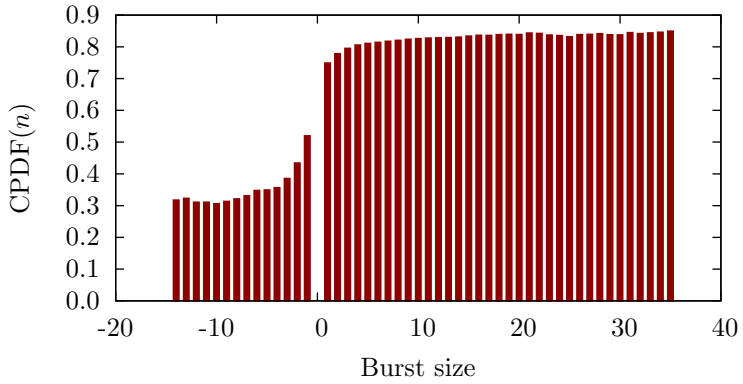


Figure 1: An example of CPDF of a link with bursty behavior.

number of consecutive frames missed. One example of a CPDF(n) from a real trace file is shown in Figure 1. As seen from Figure 1, the link exhibits a bursty behavior where the probability of successfully receiving a frame depends on the results of previous transmissions. For example, if the source node is sending two frames, the probability of correctly receiving the second frame depends on the result of the first transmission. If the first frame was correctly received, the probability of correctly receiving the second frame is 0.7484. If, however, the first frame was missed, the probability of correctly receiving the second frame is only 0.5192.

A domain of CPDF(n) of a finite trace file is a finite set since the number of consecutive frames received or missed is finite. For example, the domain of CPDF(n) of the trace file in Figure 1 is $\{-14, -13, \ldots, -1, 1, 2, \ldots, 35\}$ since the largest number of consecutive frames missed is 14 and the largest number of consecutive frames received is 35. Note that 0 is not included since the burst size 0 is undefined.

To obtain BPA(n) from CPDF(n) starting from a trace file, we first calculate the average packet reception ratio (PRR) from the trace file. The average PRR represents the probability of correctly receiving a frame when bursty characteristic of the link is not taken into account. Let $\mathbb{D}$ be the domain of CPDF(n). Let $\triangle = \max(\mathbb{D})$ (the largest number of consecutive frames received) and $\bigtriangledown = \min(\mathbb{D})$ (the largest number of consecutive frames missed). BPA(n) can be defined as:

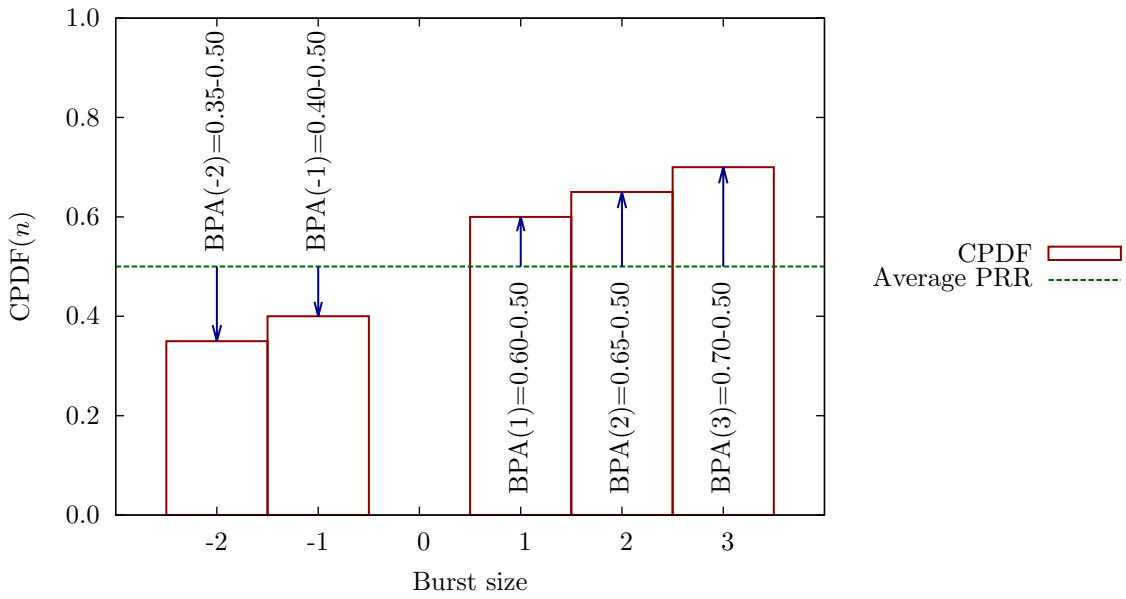


Figure 2: An example of calculating bursty probability adjustment from CPDF.

$$\text{BPA}(n) = \begin{cases} \text{CPDF}(n) - \text{PRR}, & n \in \mathbb{D} \\ 0, & n = 0 \\ \text{CPDF}(\bigtriangledown) - \text{PRR}, & n < \bigtriangledown \\ \text{CPDF}(\bigtriangleup) - \text{PRR}, & n > \bigtriangleup \end{cases}$$

We include the last two cases where $n < \bigtriangledown$ and $n > \bigtriangleup$ to handle a case when simulating a burst size larger than the bursts observed in the trace file.

An example of how to obtain BPA(n) from CPDF(n) is shown in Figure 2.

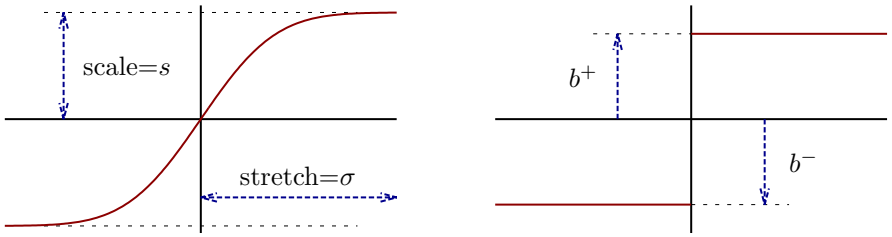
To obtain the bursty probability adjustment function, we first calculate the average PRR of the link. In Figure 2, the average PRR of the link is 0.5. The next step is to calculate CPDF(n) for different burst sizes n. The differences between the average PRR and the CPDF(n) are BPA(n). By analyzing the example CPDF(n) in Figure 2, we obtain the following bursty probability adjustment function.

$$\mathrm{BPA}(n) = \begin{cases} \mathrm{CPDF}(n) - 0.5, & n \in \{-2, -1, 1, 2, 3\} \\ 0, & n = 0 \\ -0.15, & n < -2 \\ 0.20, & n > 3 \end{cases}$$

To calculate the appropriate time-to-live for a cache entry, we use the burstiness metric (β) proposed by Srinivasan et al. [83]. The burstiness metric is a scalar value between -1 and 1 used to measure burstiness of a given link. A value close to 0 means the link is more independent (the probability does not change much when burst occurs) while a value close to -1 or 1 means that the link is very bursty (the probability changes rapidly when burst occurs). Links with negative correlation have negative β values.

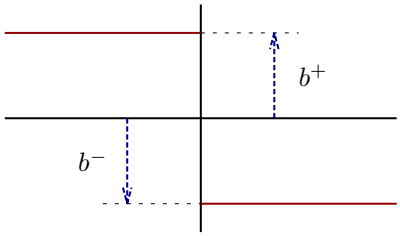
The burstiness metric of the link reduces as the duration between packets increases [83]. By decreasing the sampling rate when calculating β, the value of β decreases. To calculate the appropriate cache TTL, we decrease the β sampling rate until β is sufficiently close to 0, which means that the transmissions are almost independent of each other. The time between samples is now representing the duration where the results of previous transmissions have only a slight effect on the outcome of the next transmission. Thus, the time-to-live is equal to the time between samples.

The advantage of getting BPA(n) and TTL from a real trace file is that the bursty characteristics of the simulated link will be almost identical to the real wireless link. The drawback of the trace file-based approach is that appropriate trace files must be available. However, getting trace files may not be feasible in all scenarios, for example, when the number of nodes is large. If the network consists of N nodes, we need to obtain $O(N^2)$ trace files, which may not be feasible. We would like to be able to incorporate bursty characteristics when simulating arbitrary links.

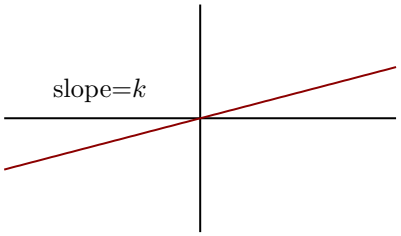


(a) scaled-error function

(b) ideal bursty link



(c) negative ideal bursty link



(d) linear bursty link

Figure 3: Examples of possible synthetic bursty probability adjustment functions.

3.2.2 Synthetic-link values

As discussed earlier, using a real trace file is the most accurate method of simulating a link. However, obtaining a trace file may not be feasible in all cases. The other method of obtaining a bursty probability adjustment function is to select one from a set of predefined functions. This method is suitable for a simulation where obtaining trace files is not possible or not feasible. The method is also suitable for a simulation where the goal is not to model links after specific real world links, but to incorporate bursty links into the simulation. We propose a set of functions in Figure 3.

Figure 3 (a) represents a BPA function that resembles the error function (erf). Figure 3 (b) and Figure 3 (c) represent the ideal bursty links where the bursty probability changes rapidly when a burst occurs. A BPA function in Figure 3 (c) shows a link with negative correlation, which has been observed in real wireless link [83]. Figure 3 (d) represents a linear bursty link where burstiness gradually increases.

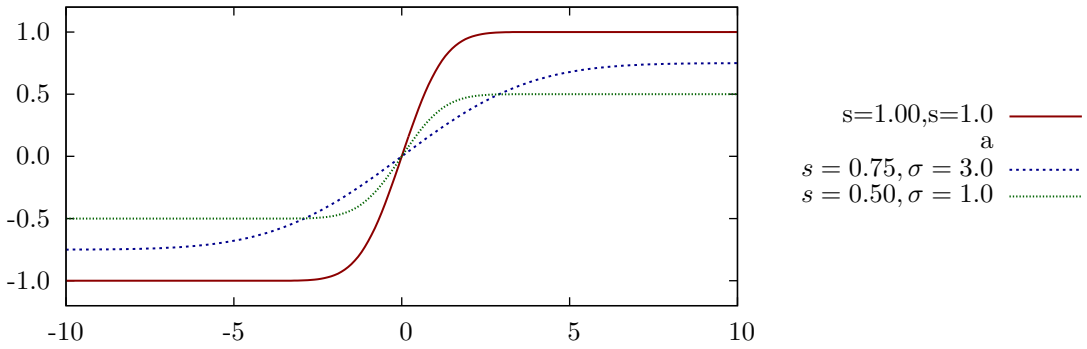


Figure 4: Examples of different scaled-erf functions.

The parameters b^+, b^- of the ideal bursty functions and the slope k of the linear bursty function, can be selected depending on the burstiness level desired. We note that there is no limitation on the actual values of the bursty probability. The only real limitation is the validity of the values and that the values returned by the model will be within a reasonable range. It is possible to use any function or a set of discrete values as a bursty probability adjustment. As mentioned previously, analyzing a trace file will result in a set of discrete values. A closed-form function, if desired, can be obtained by applying an appropriate statistical method.

The scaled-erf bursty function in Figure 3 (a) deserves special attention. We use an error function that has been *scaled* and *stretched* to obtain the bursty probability adjustment function. Our scaled-erf function takes two arguments: a scale (s), and a stretch (σ). In the standard error function, the function range is from -1 to 1. We scale the range of erf function by using s so that the range is not limited to between -1 and 1. The scaling s represents the limit to where the bursty probability adjustment function converges. We include the stretching factor, σ, to stretch the error function along the x-axis. The stretching factor changes the step sizes between bursts. The differences between two burst sizes in a function with large σ will be smaller than a function with small σ. Figure 4 shows examples of different scaled-erf functions with different parameters.

As seen in Figure 4, the effect of the scale, s, is to change the limit of the erf function. In other words, s represents the maximum probability change due to bursty behavior. The effect of the stretching factor, σ, is to stretch out the erf function along the x-axis. A large σ means that the probability differences between two burst sizes is small. For example, the probability difference between burst size 2 and burst size 1 of the standard error function is 0.2718 while the difference is 0.2338 when $\sigma = 3$.

By using s and σ to modify the shape of the error function, we can use the scaled-erf function to approximate the remaining three functions by selecting the appropriate parameters. For instance, to approximate an ideal bursty function with $b^+ = b^-$, we set s to b^+ and select a small σ such that the burst size 1 and -1 have values sufficiently close to b's (i.e. BPA$(1) \rightarrow b^+$ and BPA$(-1) \rightarrow b^-$). We can use a negative scale to get a function like the one in Figure 3 (c). For a linear bursty function, we can select a very large σ to stretch out the error function.

To get an appropriate time-to-live for a cache entry in a synthetic bursty link, we have to use a value within a range of suggested values, since we do not have a trace file to analyze like we did in Section 3.2.1. In [83], the authors suggested that the duration of correlation is usually around 500 ms. We present our own observation regarding the appropriate TTL from the trace files we generated for simulation in Section 3.4.

The advantage of using a predefined function is that it does not require a trace file. The predefined functions can be used in a network simulation with a large number of nodes where obtaining trace files is not feasible. The drawback of using a function is that the characteristics of the simulated link may not match to real wireless links.

3.2.3 Algorithm complexity

We end this section with some discussion regarding the complexity of the stochastic bursty link model.

Since the probability of correctly receiving a frame in our model is dependent on the results of previous transmissions, each node in the simulation has to keep track of transmission results from all other nodes in the network. In other words, if there are N nodes in the network, each node has to keep track of all previous transmission results from the remaining $N - 1$ nodes.

To reduce the memory requirement, the following two methods may be used. First, the stochastic bursty link model may ignore links with average PRR below a certain threshold. The model simply returns the probability of correctly receiving a frame without considering burstiness, which means that the model does not need to keep track of the history of that link. For example, the model may choose to ignore all links that have average PRR less than 0.1. In practice, applying this method means that the model will exclude all links where the two nodes are too far apart.

The second method to reduce memory requirement is to purge the history whenever it is possible. For instance, all cache entries older than the cache TTL can be removed. Moreover, the history can be purged when the transmission results switched between success or failed since a burst is defined as consecutive successful or failed frame receptions.

3.3 ns-3 implementation

In this section, we present our implementation of the stochastic bursty link model in the ns-3 simulator. We start by presenting a quick overview of the standard WiFi module in ns-3 then proceed to explain our modification to incorporate the stochastic bursty link model.

The current WiFi model in ns-3 consists of multiple components working together. The two components that are directly related to our work are `WifiChannel` and `WifiPhy` [50]. Every WiFi device has its own `WifiPhy` while a single `WifiChannel` object serves as a channel that glues all `WifiPhy`s operating in the same wireless channel together.

The main responsibility of `WifiChannel` is to pass the signal between `WifiPhy`s when transmission occurs. Given the signal, each receiving `WifiPhy` can calculate the signal-to-interference-plus-noise ratio (SINR) of the receiving packet. Packet error rate (PER) is then calculated based on the SINR and other physical layer parameters. Finally, PER is used to determine if the transmission is successful or not.

To summarize, the current steps to determine a transmission result in ns-3 WiFi module are as follows.

1. Calculate SINR of the packet
2. Calculate PER from the SINR
3. Determine if the transmission is successful

To incorporate the stochastic bursty link model in ns-3, we made changes to the default WiFi module in ns-3. We introduce a new object called `BurstyHelper` to the WiFi module. `BurstyHelper` serves as the brain of the model with two important functions. First, `BurstyHelper` acts as a memory module by keeping track of history of transmission results between all `WifiPhy`s. Second, `BurstyHelper` is responsible for readjusting the PER when burst is detected. Every `WifiPhy` holds a pointer to the `BurstyHelper` object.

At the end of the reception, `WifiPhy` first calculates the PER based on the SINR. `WifiPhy` then consults `BurstyHelper` by calling `ReadjustPer`. `BurstyHelper` looks at the cache of previous transmissions from the sending `WifiPhy` to the receiving `WifiPhy` to see if there was a burst prior to the current packet. If there was a burst,

`BurstyHelper` adjusts the PER by consulting the bursty probability adjustment function and returns the new PER to `WifiPhy`. `WifiPhy` then determines the result of the reception using the adjusted PER value. Finally, `WifiPhy` reports the result of the reception to `BurstyHelper` so that `BurstyHelper` can update the burst size.

Overall, the following changes were made to WiFi module.

1. A new object called `BurstyHelper` is included in the WiFi module
2. `BurstyHelper` keeps track of transmission history between all pair of `WifiPhy`s
3. `WifiPhy` stores a pointer to the `BurstyHelper` object
4. `WifiPhy::EndReceive` now takes a pointer to the sending `WifiPhy`
5. `WifiPhy` calls `BurstyHelper` to check if any adjustment to the default PER is required
6. `WifiPhy` determines the outcome of the reception and reports the result to `BurstyHelper`

The new steps to determine a transmission result in the modified WiFi module are as follows.

1. Calculate SINR of the packet
2. Calculate the default PER from the SINR
3. Look at the history of the transmissions
4. Adjust the PER if necessary
5. Determine if the transmission is successful
6. Save the result of the transmission

3.4 Evaluation

We evaluate our stochastic bursty-link model by using the implementation in ns-3. We evaluate our stochastic bursty-link model in three different aspects. First and most importantly, we evaluate how well our model is able to replicate the bursty behavior observed in real wireless links. Second, we study how well our stochastic bursty-link model can simulate real wireless links with different bursty characteristics. Our goal is to study the difference between using a discrete BPA function and using a synthetic function to simulate wireless links. Finally, we study how our stochastic bursty-link model affects the routing protocol performance.

3.4.1 Trace files generation

As stated earlier, the most important goal of our model is to replicate bursty characteristics observed in real wireless links. To evaluate our model against real wireless links, we obtained trace files from an indoor office environment. We used three laptops and one desktop to gather traces files. BackTrack 5 R1 was installed on all devices. One laptop equipped with a wireless adapter based on the Atheros chipset [6] was selected as a packet sender. An application that continuously broadcasts UDP packets at the rate of 100 packets per second using 802.11g was installed on the packet sender. Each UDP packet contains a unique packet identification number for later analysis. Transmission power and data rate were kept constant throughout the trace files collection using the command `iwconfig` [7]. The remaining machines were used to capture the packets. One capturing laptop was equipped with Atheros-based wireless adapter while the other laptop was equipped with a RaLink-based wireless adapter [3]. The capturing desktop was equipped with a RaLink-based wireless adapter.

We created trace files using `tcpdump`. All wireless interfaces were switched to monitor mode with `airmon-ng` [1]. The trace files contain information about the received signal strength of each frame provided by the Radiotap header. All captures

were done within the Klaus Advance Computing Building on the Georgia Institute of Technology campus with a duration of one hour per trace file. We obtained 100 trace files for the simulations. Full information regarding the 100 trace files can be found in Appendix A.

3.4.2 Evaluation against real wireless links

We evaluate two variations of our stochastic bursty link model: using discrete BPA functions directly analyzed from trace files and using a scaled-erf BPA function. We used `MatrixPropagationLossModel` as a base propagation loss model for both variations. We set the loss such that the average PRR of the link is equal to the PRR of each trace file. For the discrete BPA function, we followed the steps described in Section 3.2.1 to obtain the BPA function. First, we calculated cache TTL for each trace file by decreasing β sampling period until $|\beta| < 0.1$ and then used the TTL as a parameter to our model. Finally, the discrete BPA function of each trace file was obtained by analyzing the CPDF of the trace file.

For the scaled-erf BPA function, we manually tuned the two parameters of the scaled-erf function (s and σ) to match with the shape of the CPDF from the trace file. To obtain the TTL, we analyzed the 100 trace files. We varied the β sampling period of the 100 trace files and looked at the distributions of $|\beta|$'s. Complementary cumulative distribution function of the $|\beta|$ values of the trace files with different sampling period are reported in Figure 5.

As seen from Figure 5, at the sampling period of 10 ms (counting every packet), all 100 trace files had $|\beta| > 0.1$. When the sampling period decreased to 100 ms (counting every 10 packets), the number of trace files with $|\beta| > 0.1$ dropped to 28. At the sampling period of 500 ms (counting every 50 packets), only 4 trace files still had $|\beta| > 0.1$. Thus, we used the TTL of 500 ms for the scaled-erf BPA function. The value of 500 ms is also in agreement with the previously suggested value [83]. Note

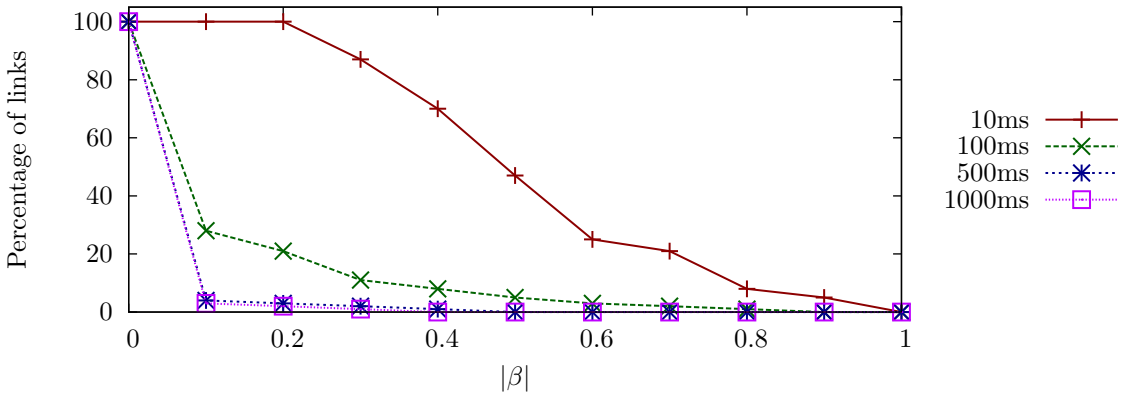


Figure 5: Complementary cumulative distribution function of $|\beta|$ values.

that this TTL is used in *all* scaled-erf BPA functions. In the case of discrete BPA function, the TTL is different for different trace files.

We compared the results of our model with the trace file and two other models: the default ns-3 model and BEAR [32]. For the default model, we used the `MatrixPropagationLossModel`. We set the loss such that the average received signal strength is equal to the average received signal strength of each trace file. We added a fading effect by using a `RandomPropagationLossModel` with a `NormalRandomVariable` with mean 0 and variance corresponding to the trace file. For BEAR, the parameters were tuned using the received power traces from the trace files with the order of 3.

We presented the simulation results by using the β values. We selected a representative subset of the trace files with varying β values. All results from simulated links were averaged from 1000 simulations. The simulation results are reported in Figure 6. The full simulation results from all 100 trace files can be found in Appendix B.

As seen from Figure 6, our model can simulate burstiness of the trace file from very bursty links ($\beta \to 1$) to almost independent links ($\beta \to 0$). The β values of both the discrete BPA function variations and the scaled-erf BPA variations function differ from the trace files' only slightly. The default LogDistance with fading model in ns-3 does not exhibit any bursty behavior as the β values are close to 0; that highlights its memory-less behavior. We show the detailed results from one of the trace files

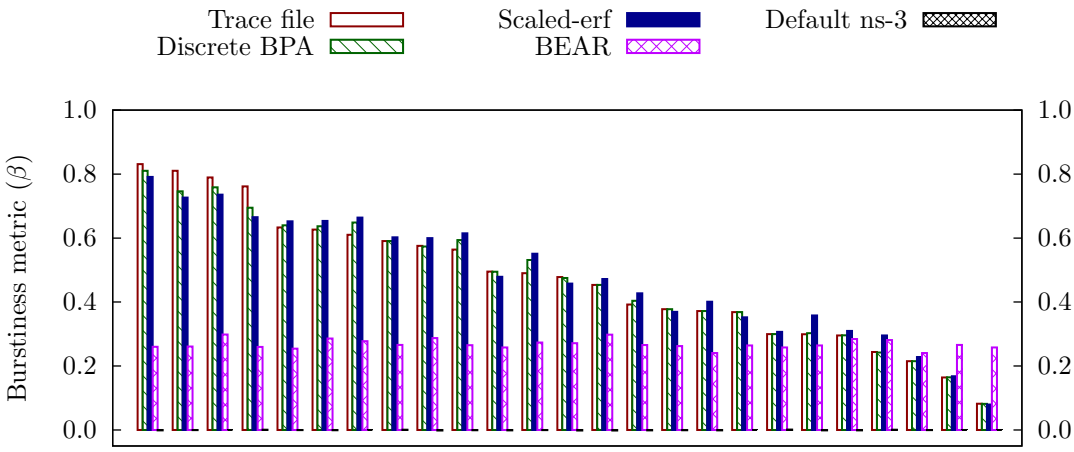


Figure 6: A comparison between simulated links and trace files.

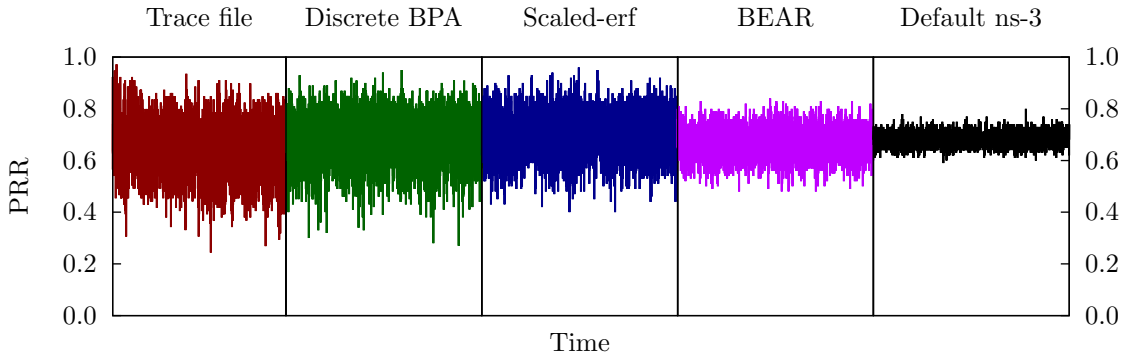


Figure 7: Packet reception ratio for the trace file and different models.

in Figure 7, Figure 8, and Figure 9. Figure 7 shows the PRR of the trace file and different simulated links. Figure 8 shows the CPDF of the trace file and the simulated links. Figure 9 shows the distribution of burst sizes.

In this set, the average packet reception ratio of the trace file is about 0.67. As seen from Figure 7, the PRR of the default model remains relatively stable when compared to the trace files and other models. Figure 8 confirms that the CPDFs of both variations of our model closely resemble the CPDF from the trace file while the CPDF of the default model is almost flat. In other words, the randomness of the default model does not capture the bursty effect observed in a real network. Figure 9 shows the distributions of burst sizes of the trace file and the simulated links. The

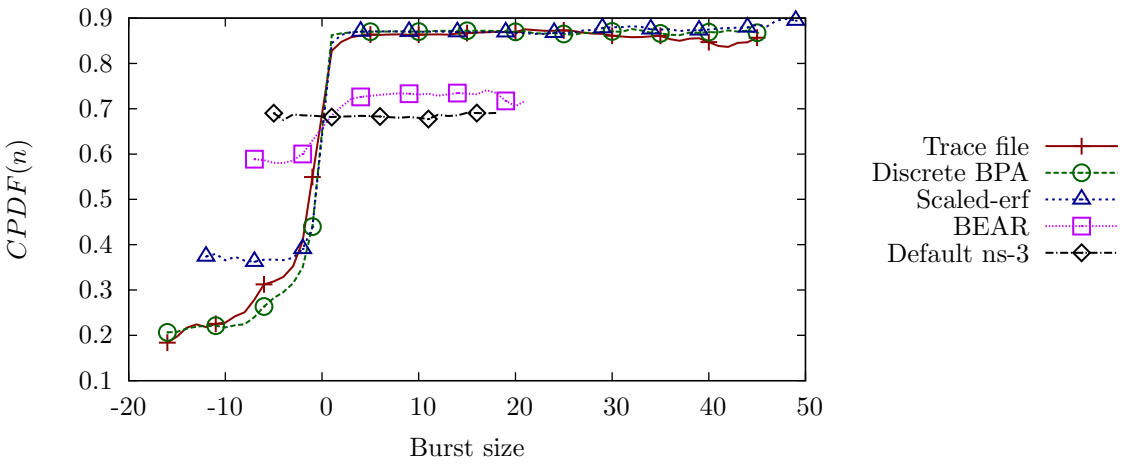


Figure 8: CPDF from one trace file along and its simulated links.

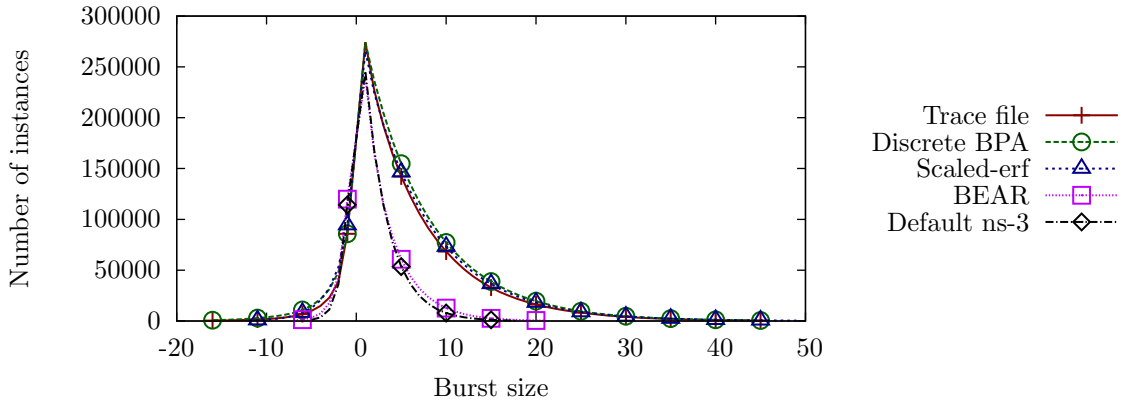


Figure 9: Distribution of burst sizes from the trace file and different models.

distribution of the burst size of our model is almost identical to the trace file while the distributions of burst sizes of the default model and BEAR are noticeably differ from the trace file. Note that the number of instances shown in Figure 9 are cumulative (e.g. burst size 3 is also counted in burst size 4).

The β values of BEAR show that it can simulate a bursty link to a certain degree. However, the BEAR model is slower to change when compare to our model since it uses an auto-regression model to simulate links. Moreover, BEAR will not be able to simulate bursty links when the cause of the burstiness is not the variation of the received powers. We show packet reception ratio along with average received power of one trace file in Figure 10.

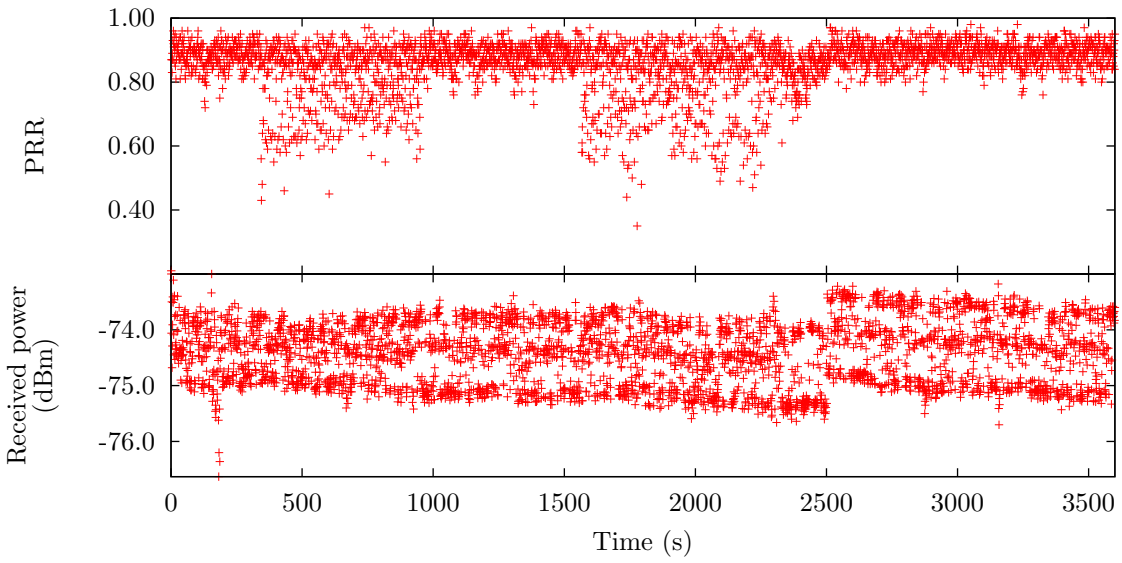


Figure 10: Average packet reception ratio and received power trace.

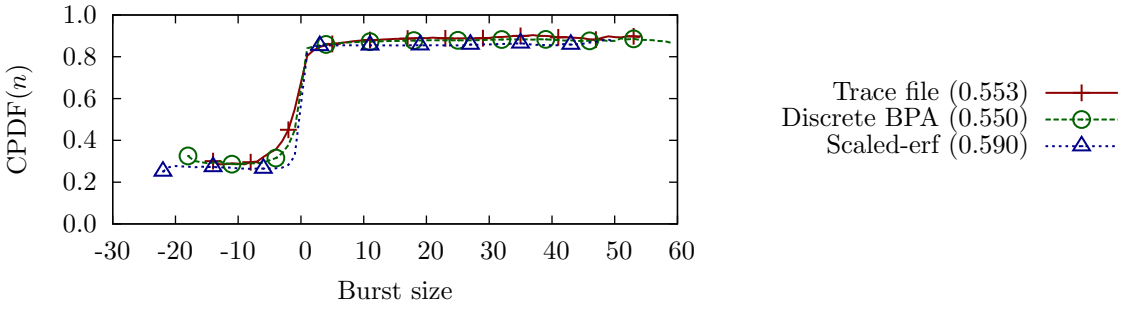
As seen from Figure 10, the link exhibits bursty behavior where the packet reception ratio dropped between 400 s to 1000 s and between 1600 s to 2000 s. However, the received power trace does not show any substantial drop during those time periods.

3.4.3 Simulating links with different bursty characteristics

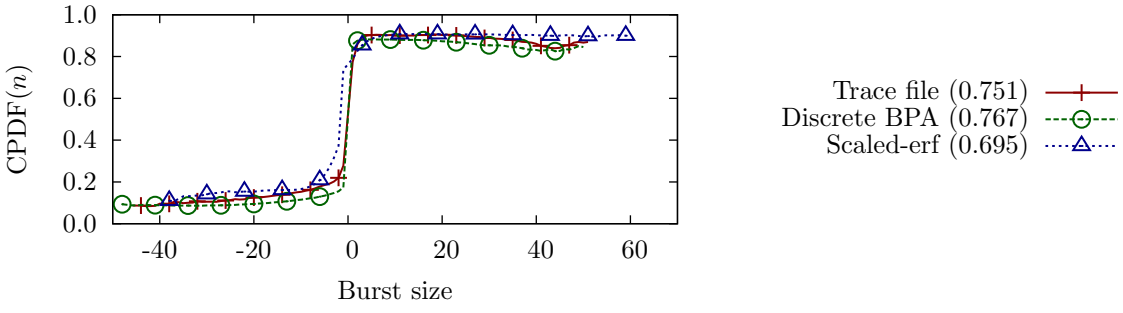
In this section, we compare between the two variations of our model. Specifically, we compare between using a discrete BPA function from a trace file and using a scaled-erf function. We would like to see how well the scaled-erf function simulates different CPDFs from trace files when compared to directly using discrete BPA functions from trace files.

To compare between the two methods, we selected four trace files with different bursty characteristics and show the results from simulations with discrete BPA functions and scaled-erf functions. The four sets of CPDFs are reported in Figure 11.

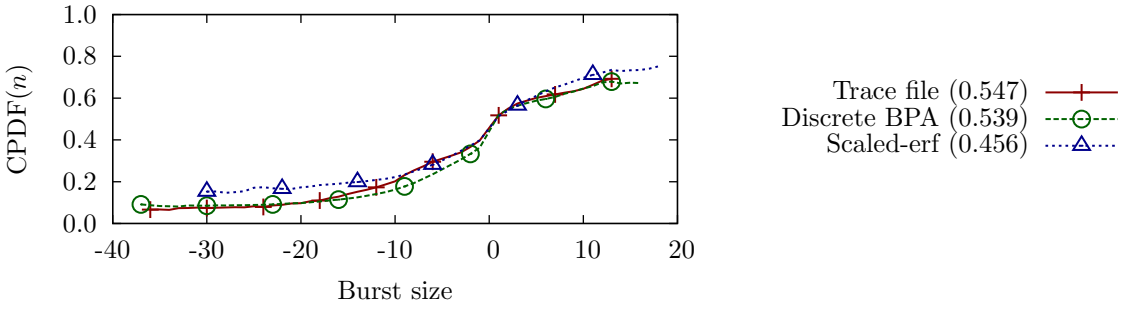
In Figure 11 (a), the trace file shows a slightly bursty behavior with small burstiness metric. Figure 11 (b) shows an example of a very bursty link where the probability shifts quickly depending on the history of the link. In Figure 11 (c), the link shows a slightly bursty behavior with different shape from the link in Figure 11 (a).



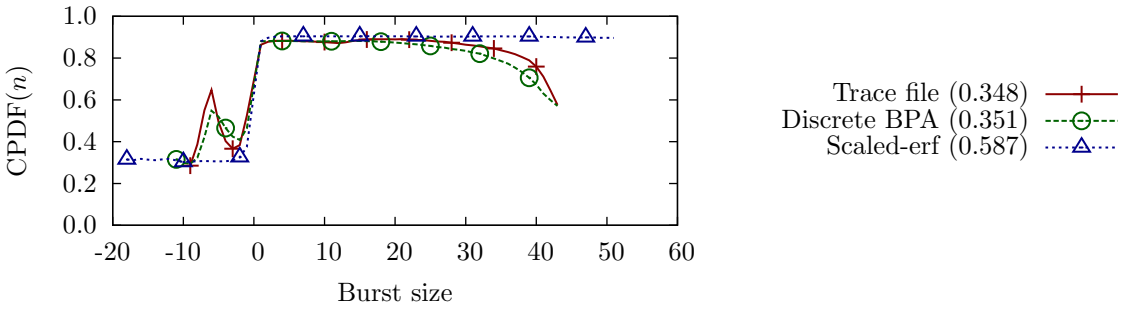
(a) a link with moderate burstiness



(b) a link with an almost ideal burstiness



(c) a link with an almost linear burstiness



(d) a link with abnormal CPDF shape

Figure 11: Using a scaled-erf function to simulate links with different burstiness.

As shown in Figure 11, our model with discrete BPA functions can simulate links with different bursty behaviors. The burstiness metrics of the discrete BPA functions are almost identical to the trace files. The scaled-erf model can simulate the links in Figure 11 (a), (b), and (c) well but cannot simulate the link in Figure 11 (d) due to its strange CPDF shape. The shape of the CPDF in Figure 11 (d) does not resemble any scaled-erf function. In this case, the scaled-erf function performs poorly while the discrete BPA function is still able to mimic the strange CPDF shape.

As noted earlier, one drawback of using a scaled-erf function instead of a discrete BPA is that, the function may not be able to simulate some CPDF shapes. The shapes of the CPDF simulated from the scaled-erf function will be smoother than the real trace files as the function is not discrete. Thus, a discrete BPA function may be needed to simulate links with strange CPDF shapes.

3.4.4 Effect of the simulated bursty link on routing protocols

In this section, we investigated the effect of our stochastic bursty-link model on routing protocols. Our goal was to study the effect on routing protocols when the bursty behavior is incorporated into the wireless simulation. We studied two scenarios with routing protocols: a diamond topology and a random network. First, we studied the diamond topology since the topology is simple and the effect of incorporating the stochastic bursty-link model can be easily observed. Next, we proceeded to study the random topology.

3.4.4.1 Diamond topology

We began the study with a network with four nodes placed in a diamond topology as shown in Figure 12. There is one source node s and one destination node t. Node u and v serve as forwarding nodes for routes between s and t. We manually set the propagation loss in the network such that there are only four possible links in the topology: $s \leftrightarrow u$, $s \leftrightarrow v$, $u \leftrightarrow t$, and $v \leftrightarrow t$. The loss on the four links, ϕ, can be set

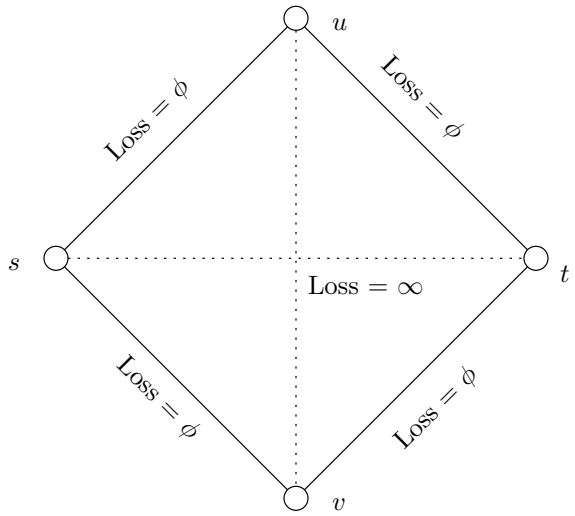


Figure 12: A diamond topology used to study performances of different models.

to different values to get different frame success rates. Other links were configured with infinite loss. To accomplish this, we used `MatrixPropagationLossModel` in ns-3 as the base propagation loss model. The topology was selected to ensure that s must reach t by using a routing protocol.

We compared the results between three models: the default ns-3, BEAR, and our stochastic bursty link model. Fading effect was added to the default ns-3 model using the `RandomPropagationLossModel`. For our model, we evaluate three variations:

1. "Ideal bursty" – ideal bursty link ($b^+ = b^- = 0.2$),
2. "Positive" – positive burst only ($b^+ = 0.2, b^- = 0$), and
3. "Negative" – negative burst only ($b^+ = 0, b^- = 0.2$).

The positive burst only model is a model where the probability of correctly receiving a frame increases when a positive burst occurs and the probability resets to the default value when a negative burst occurs. In other words, the quality of the link in the positive burst only model is always at least as good as the default link. For the negative burst only model, the link is always at most as good as the default link. The

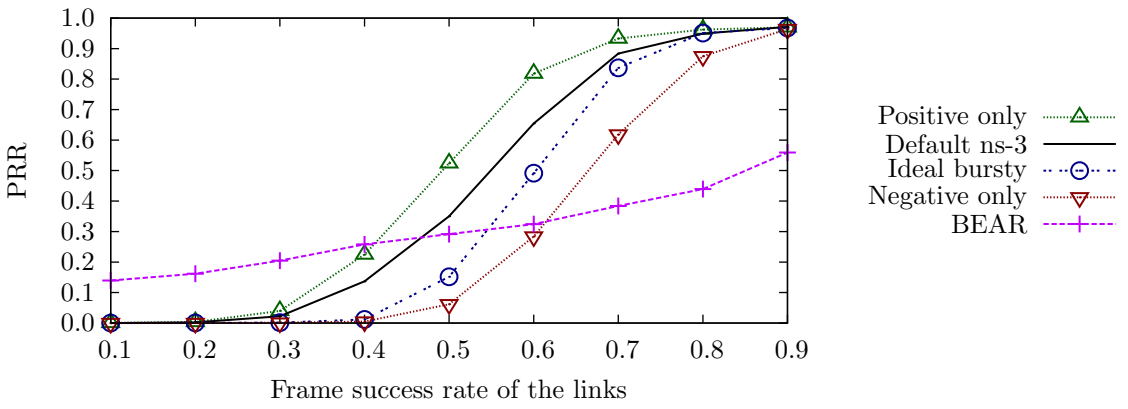


Figure 13: Packet reception ratios of different models with varying link quality.

positive burst only model and the negative burst only model are not representatives of real wireless link, but are included for comparison purposes.

To observe the behavior of different models under different network conditions, we varied ϕ such that the frame success rate of the links is between 0.1 (very bad links) and 0.9 (very good links). The UDP application on the source node generates packets at the rate of 20 packets per second. We reported simulation results using packet reception ratio. All results reported are averaged from 1000 simulations. The simulation results are reported in Figure 13. For better readability, the confidence intervals are not shown since the intervals are very small.

As seen from Figure 13, packet reception ratios of all models increase as the link quality improves, which is expected. The positive burst only model has higher PRR than the default ns-3 since the quality of the link increases when positive bursts occur but the link quality never drops below the default values. On the other hand, the link quality of the negative burst only model is always lower than the default ns-3, resulting in lower PRR than the default model. However, the effect of positive burst only and negative burst only are *not* equivalent. The results show that negative bursts have more impact on the PRR than positive bursts. This behavior is expected since DSR takes time to find a new route to the destination when the current route is broken. From DSR's perspective, there is not much benefit from a positive burst other than

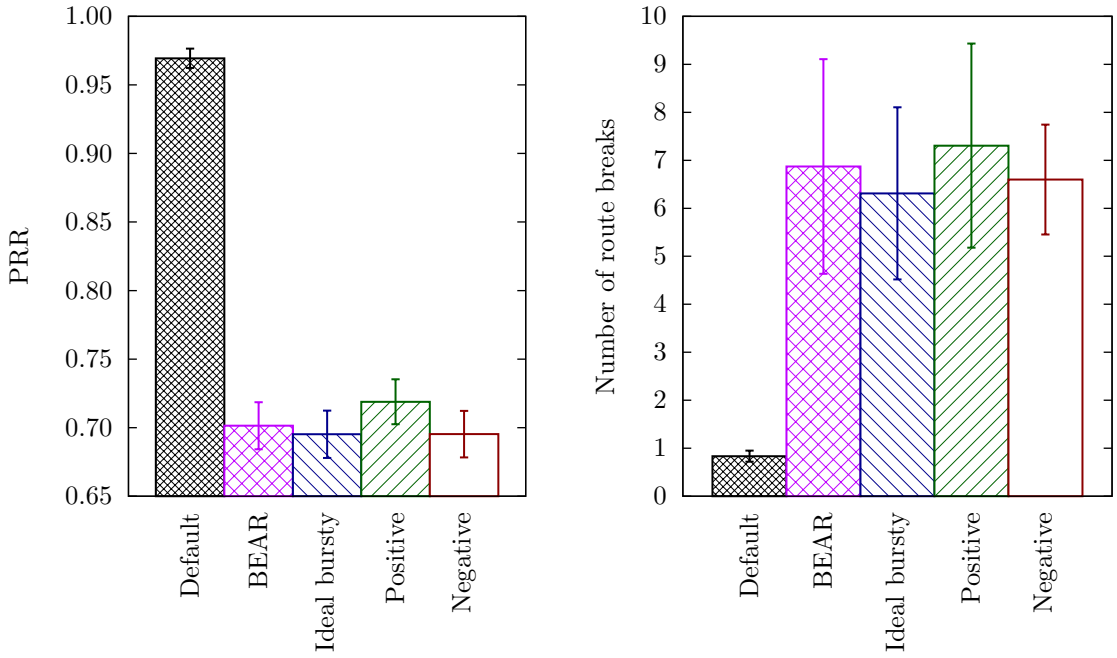
shorter delivery delay (no retransmission is required at the MAC layer). However, when a negative burst occurs, a DSR route may break and must be reestablished. This behavior can be observed in the ideal bursty link model ($b^+ = b^-$) where the effects of negative bursts outweigh the effects of positive bursts.

3.4.4.2 Random topology

Finally, we turn our attention to a random network. In this study, we randomly place 50 static nodes in the deployment area of 500 m by 500 m. Two nodes are selected as a source-destination pair where the distance between the two nodes is at least 300 m to ensure that the two nodes use routing to reach each other. The application on the source node generates packets at the rate of 20 packets per second for 480 seconds. We ran two sets of application: one with UDP and one with TCP.

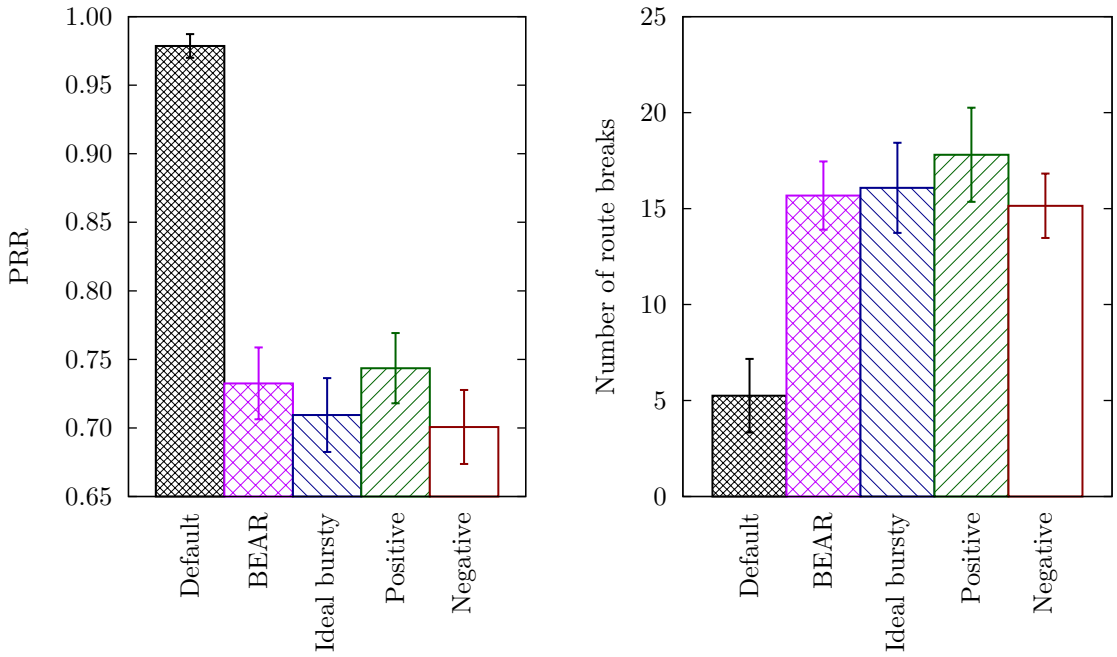
Again, we compare the results between three models: the default ns-3 model, BEAR, and the three variations of our bursty link model. For the base propagation loss model, we use the `LogDistancePropagationLossModel` provided by ns-3. We report the simulation results in two aspects: the packet reception ratio and the number of route breaks. All simulation results are averaged from 1000 simulations and reported with 95% confidence interval. UDP simulation results are reported in Figure 14 and TCP simulation results are reported in Figure 15.

As seen in Figure 14 and Figure 15, the default model has the highest packet reception ratio with almost 100% delivery. All models with memory effect have substantially lower packet reception ratios. The lower packet reception ratios of other models result from more frequent link breakage during the simulations. The simulation results show that the performance of DSR is significantly different when bursty behavior is incorporated into the simulation. All models with memory experienced about 4 to 7 link breaks with UDP application and about 14 to 15 link breaks with TCP application. The default ns-3 model experienced only about 1 link break with



(a) packet reception ratio (b) number of route breaks

Figure 14: DSR simulation results with different bursty link models (UDP).



(a) packet reception ratio (b) number of route breaks

Figure 15: DSR simulation results with different bursty link models (TCP).

UDP application and about 4 link breaks with TCP application. The more frequent route breaks mean that DSR has to reinitiate the route discovery process more often. The smaller number of route breaks of the default ns-3 model results in higher PRR but the results are not realistic. This is in general agreement with published results from DSR in a real wireless ad hoc network testbed, which showed that route breaks occurred much more frequently even in a well-planned network as small as two hops [59, 60].

3.5 Discussion

In this chapter, we have proposed a new stochastic frame-level bursty-link model for wireless network simulation. Our model simulates bursty behavior of wireless links by adjusting the probability of correctly receiving a frame based on the history of the wireless link. Our model can directly simulate a real wireless link by modeling the bursty characteristics from the trace file or simulate a bursty link by using predefined functions. We have implemented the model in ns-3 simulator and showed that our model is able to replicate bursty behavior observed in real wireless links. We also comprehensively studied the effect of using our stochastic bursty-link model on the routing protocol performance and showed that the routing protocol performance was significantly affected by the bursty behavior of the wireless links.

CHAPTER IV

LINK DISCOVERY WITH RELIABLE MULTICAST PROTOCOL

In this chapter, we consider the problem of multicasting at the MAC layer. We proposed an extension for IEEE 802.11 MAC layer that provides reliability for multicast transactions and incorporates a neighborhood maintenance mechanism. Our proposed protocol, called link discovery with reliable multicast protocol (LDM) uses positive acknowledgment to provide reliability for multicast transactions. LDM also includes a mechanism for a node to quickly detect a new neighbor that moves into its range without relying on external mechanism.

4.1 Link discovery protocol and reliable multicast

We propose an extension for the IEEE 802.11 framework called link discovery with reliable multicast protocol (LDM). The proposed protocol has two main goals. One goal is to provide reliability for MAC-layer multicast frames and the other is to dynamically track nodes' neighbor sets within the MAC layer. To achieve the first goal, LDM uses positive acknowledgement mechanism. To achieve the second goal, LDM provides a mechanism for devices to quickly recognize changes in their neighbor sets. Since many higher-layer protocols require neighborhood maintenance, supporting this capability efficiently at the MAC layer will both simplify the design of higher layers and eliminate potential redundancies in their execution. Thus, a unified MAC layer that supports both reliable multicast and neighborhood maintenance will streamline overall network performance. We present the reliable multicast protocol it relies on, before presenting our link discovery mechanism.

4.1.1 Basic reliable multicast protocol

LDM uses positive acknowledgement mechanism to provide reliability for multicast transactions. LDM modifies the default 802.11 frame headers to include additional receivers' addresses for a multicast transaction. Figure 16 shows the modified frame structure. LDM introduces a new field in the MAC header called Extended Control field. The Extended Control field is an 8-bit field where the least significant bit is called the "Join ACK" bit, and the next three bits are called the "Join ACK level".

Similar to 802.11 unicast, LDM supports both two-way and four-way transactions. LDM differentiates between two-way and four-way transaction by frame size. Figure 17 illustrates the two scenarios of LDM.

For a two-way transaction, the DATA frame is modified to include multiple receivers addresses. The ACK frame is modified to include the ACK sender's address. The multicast source selects as many receivers from its neighbor list as permitted by the maximum 802.11 frame size. Thus, the total number of addresses LDM can have in a two-way transaction is limited by the data size. The multicast source splits a multicast transaction into multiple subtransactions if it cannot support all neighbors in one transaction. The multicast source sets the Join ACK bit to 1 in the last subtransaction, and to 0 in the other subtransactions.

A multicast source initiates a two-way transaction by sending a modified DATA frame. If the multicast source selects N receivers and sets the Join ACK bit to 0, the time after the DATA frame is divided into N ACK slots. If the Join ACK bit is set to 1, the time is divided into $N+1$ ACK slots. All ACK slots are separated by SIFS. When a node receives the DATA frame, it checks if the DATA frame is addressed to itself or not. If the DATA frame is addressed to itself, the node schedules transmission of a modified ACK frame in a corresponding ACK slot according to its position in the DATA frame. If ACK frames from all N selected neighbors are received, the multicast source considers the multicast transaction completed. If ACKs from some receivers

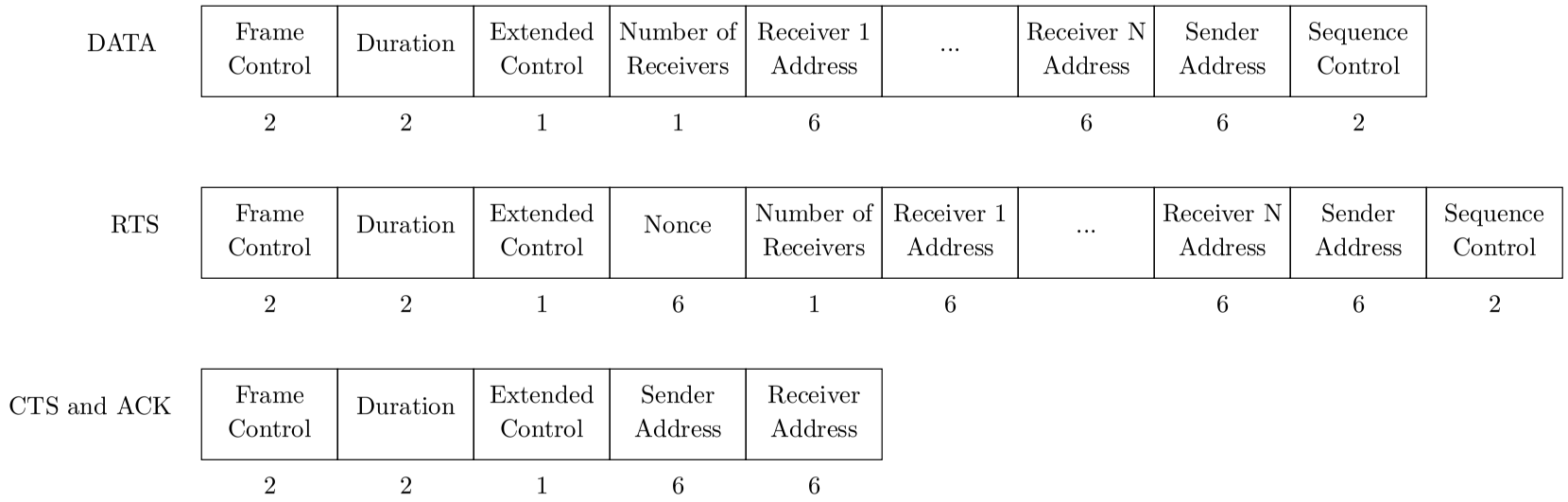
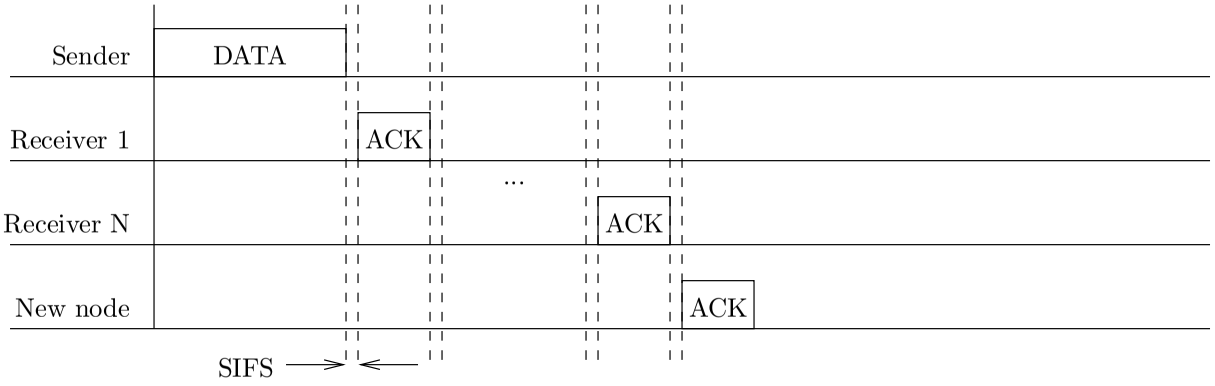
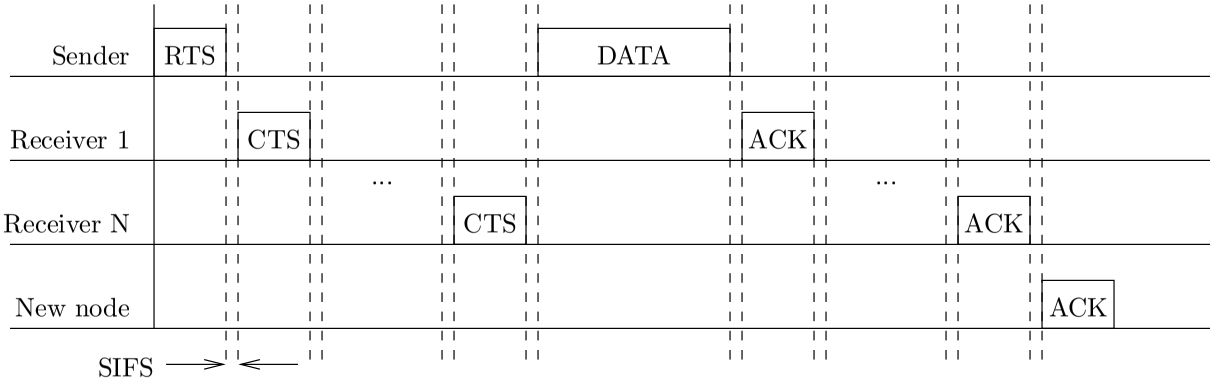


Figure 16: Modified 802.11 headers used by LDM protocol.



(a) Two-way transaction (frame size < threshold).



(b) Four-way transaction (frame size ≥ threshold).

Figure 17: Two types of transaction of LDM protocol.

are not received, the source reincludes the missed-ACK receivers in a subsequent subtransaction or restarts the multicast transaction for the missed-ACK receivers if no additional subtransactions are scheduled. The source retransmits to each failed receiver seven times before giving up.

For a four-way transaction, the RTS is modified to include multiple addresses and an 48-bit nonce. A multicast source initiates a four-way transaction by sending a modified RTS frame that includes N selected receiver addresses and an 48-bit nonce. The time after the RTS frame is divided into N CTS slots. When a node receives the RTS frame it checks if its address is present in the RTS frame or not. If the node address is included in the RTS frame, it schedules transmission of a modified CTS frame according to its position in the RTS frame. All nodes that received the RTS frame save the 48-bit nonce associated with the RTS frame.

If the multicast source receives at least one CTS, the multicast source schedules transmission of the DATA frame. The DATA frame is a standard 802.11 DATA frame with the address `FB:FF:FF:FF:FF:FF` in the Address 1 field to indicate that the DATA is the multicast frame. The DATA frame also has the previously generated nonce in the Address 3 field. The purpose of the 48-bit nonce is to match between RTS and DATA frames. The time after the DATA frame is divided into N ACK slots if the Join ACK bit was set to 0 or $N + 1$ ACK slots if the Join ACK bit was set to 1. Each multicast receiver that correctly received the DATA frame with the matched nonce schedules transmission of a modified ACK frame in an ACK slot according to its position in the RTS frame. If all ACK from N selected receivers are received, the multicast source considers the multicast transaction completed. If ACK frames from some receivers are missing, the multicast source reincludes the missed ACK receivers in the subsequent transactions.

4.1.2 Link discovery

Our second goal in designing LDM is to enable nodes to quickly recognize neighborhood changes and to eliminate the need for a separate neighborhood maintenance mechanism. Neighborhood maintenance typically involves the use of separate HELLO messages in a network. This HELLO mechanism, in addition to being wasteful of network bandwidth [31], has several other deficiencies we demonstrate in Section 4.2. In LDM, every frame that includes its sender's address and has the same transmission characteristics as a DATA frame serves the same function as a HELLO message. Every node has a countdown timer that counts from the last time it sent a frame that can be treated as a HELLO message. If the countdown timer expires, the node sends out a data frame that serves as a HELLO message.

In our protocol, we distinguish between *incoming* neighbors and *bidirectional* neighbors. Node A considers node B as an incoming neighbor if A received frames transmitted by B. For node B to be considered a bidirectional neighbor by A, the following two conditions must be satisfied

1. B must be able to receive frames sent by A, and

2. A must be able to receive ACKs from B.

We do not assume that all links are bidirectional as Kotz, et al. [47] showed that the probability of an asymmetric link can be as high as 24 percent. Assuming all links are bidirectional can degrade the network performance when unidirectional links are present [17, 24, 46, 61, 72, 73, 100, 102].

To enable fast neighbor discovery and differentiation between unidirectional and bidirectional links, LDM provides an extra $(N + 1)$th ACK slot for a new node to send an ACK frame called the Join ACK slot. If the node is not addressed in the DATA/RTS frames and the Join ACK bit is set to 1, the node randomly decides to send an ACK in the Join ACK slot with a probability that is indicated by the Join

ACK level. Therefore, the new node is able to make its presence known to the sender as soon as it receives a DATA frame. The sender can also classify the new node as a bidirectional neighbor immediately since both bidirectional conditions are satisfied.

The Join ACK level maps to a probability value, which is used to reduce the chance of ACK collision when multiple nodes try to join at the same time. The sender adjusts the level according to what happened in the previous Join ACK slot. The sender assumes that the probability is too low if no transmission is detected during the previous Join ACK slot, and increases the probability level. If the sender detects a transmission, but failed to receive a frame, it assumes that multiple nodes are trying to join at the same time. The sender then decreases the probability level. If the sender correctly receives an ACK from a new neighbor, the sender does not change the probability level. The mapping between the probability level and the probability value, and how a sender adjusts the probability level, can be set to match a current network's condition.

Neighbor classification works as follows: consider two nodes A and B; node A classifies node B as an incoming neighbor if node A receives a DATA frame or a HELLO from node B that does not include node A in the destination list. Node A then sends a Join ACK to node B. After receiving a Join ACK from node A, node B classifies node A as a bidirectional neighbor, because node B is certain that its transmission can be ACKed by node A. In the next transmission by node B, it includes node A in the destination list. Upon receiving the DATA from node B, node A can now classify node B as a bidirectional neighbor since node A knows that its Join ACK was correctly received by node B. If a link from node B to node A is unidirectional, node A will recognize node B as an incoming neighbor since it can receive a DATA frame or a HELLO from node B. In this case, node A keeps track of the number of times it sends a Join ACK to node B and it stops trying to join after seven attempts, at which time it classifies the link as unidirectional.

Two mechanisms are used to detect when a neighbor leaves the neighborhood: a retransmission limit and a timeout. A node keeps track of the number of retransmissions to each bidirectional neighbor. If the number of retransmissions is seven, the node considers the neighbor to be an incoming neighbor. A node removes an entry from its neighbor list if it has not received any frame from a neighbor within a timeout period.

4.1.3 Reliability and scalability

LDM uses positive acknowledgments as a means to provide reliability. A sender includes a list of all intended receivers in an RTS or a DATA frame. Each receiver then replies with a CTS or an ACK sequentially according to its position in the RTS or the DATA frame. If a neighbor did not reply with an ACK, the source resends to that neighbor in subsequent subtransactions or in a separated transaction.

As illustrated in Figure 17, the transmissions of CTS and ACK from receivers one by one are time consuming. The time required is an increasing function of the number of multicast receivers. Although we do not set a limit on the number of multicast receivers, there are two factors that affect the maximum number of receivers in the multicast transactions.

First, the IEEE 802.11 specification imposes limits on the maximum frame size. Thus, the maximum number of addresses in the header depends on the size of the data. LDM is guaranteed to support at least three receivers since the original MAC header has four address fields. LDM uses one address field for the sender address and the remaining three address fields for three receivers. More than three receivers can be supported if the data size is smaller. If a hard limit is placed on the data size, the minimum number of receivers that can be supported in one frame can be increased.

The second factor is the overhead of the positive acknowledgement approach. One of the problems of using the positive acknowledgement approach is the ACK explosion

problem. All multicast protocols that employ the positive acknowledgement approach experience this problem. We evaluate the efficiency of the protocol in Section 4.2.8.

4.2 Performance evaluation

We have evaluated our protocol performance through simulation. In this section, we provide details of the simulation environment, the assumptions, and the simulation results.

4.2.1 Simulation parameters and assumptions

We used ns-3.10 simulator to evaluate the performances of all protocols. We considered the physical interference (PI) model in this work [36]. In the PI model, interference from all concurrent transmitters in the network, no matter how distant, is factored into the signal-to-interference-plus-noise ratio (SINR) value at the receiver, and the SINR value determines the probability that a transmission is successful.

We compare our protocol against existing 802.11 reliable multicast protocols, MMP [34] and MWB [40], supplemented with HELLO-based neighborhood maintenance. All HELLO messages have the same characteristics as DATA frames [16]. Two variations of HELLO mechanisms were used: a simple HELLO message mechanism where all nodes send a HELLO message every one second and the timeout is two seconds, and the TAP protocol [39], where HELLO rate varies with node velocity.

For LDM protocol, the Join ACK probability value ranges from 0.125 to 1. The mapping function used in the simulation was $\Pr(Join) = \frac{1}{8} \cdot (1 + L)$ where L is the Join ACK level that ranges from 0 to 7. If a multicast source does not hear any transmission during the previous Join ACK slot, L is increased by one. If a multicast source detects a transmission but fails to receive the frame in the previous Join ACK slot, L is decreased by one.

Common simulation parameters in Table 1 are used in all simulations, unless stated otherwise. All simulation results are averaged over ten simulation runs.

Table 1: Simulation parameters used to evaluate MAC layer multicast protocols.

Parameter	**Value**
Deployment area	1000 m by 1000 m
Mobility model	random waypoint [98]
Speed	v to $v + 2$ m/s
Pause time	0 seconds
Simulation duration	600 seconds
Propagation loss model	log-distance
Path-loss exponent	3
Device	IEEE 802.11g [8]
Transmission power	30 mW
RTS/DATA	54 Mbps
CTS/ACK	24 Mbps
Application	on-off
Application data rate	2.5 Mbit/s
Application on duration	1 second
Application off duration	$U(1, 2)$ seconds
Packet size	$U(128, 1920)$ bytes
RTS threshold	1024 bytes

4.2.2 The idealized neighborhood relationship

Under the PI model, the relationship between SINR and the probability of successful transmission is not a step function, where a transmission is always failed when SINR is below a certain threshold, and always successful when SINR is above this threshold. Since there is no threshold SINR in the PI model, there is no set maximum distance between transmitter and receiver and hence, the definition of which nodes are neighbors at any particular point in the simulation is not obvious. To evaluate how well the protocols maintain neighborhood information, we define an *idealized* neighborhood relationship between two nodes. Ideally, two nodes are considered neighbors if the distance between them maps to a specific SNR value or higher. The definition of the

ideal neighbor distance is introduced as a means to evaluate how well the protocols maintain neighborhood information, but it does *not* affect the behavior of the protocols. In the simulations, a node considers another node as a bidirectional neighbor if it recognizes that there is a bidirectional link between them.

4.2.3 Evaluation metrics

We evaluated two aspects of each protocol: neighborhood maintenance and reliability. To evaluate a protocol's ability to maintain neighborhood information, we added two metrics to ns-3. The first metric was used to record *neighbor add delay*, which is the difference between the time when a node recognizes a new neighbor and the time when the new neighbor actually moves within the ideal neighbor distance. If the node recognizes the new neighbor before the new neighbor moves within range, which is possible due to the idealized definition of the neighbor distance, the neighbor add delay is set to zero.

The second metric was used to measure a protocol's ability to maintain neighborhood information in an environment where links' states are constantly changing between bidirectional and unidirectional. The second metric counts the number of transmissions resulting from a node that incorrectly classifies a unidirectional neighbor as a bidirectional neighbor. These transmissions waste bandwidth since the node is expecting an ACK from a unidirectional neighbor.

Multicast packet reception ratio (MPRR) was used to evaluate the protocol's reliability. Packet reception ratio for one receiver is defined as the total number of bytes received by that receiver divided by the number of bytes sent by a source during the time that the receiver was inside the ideal neighbor range from the source. MPRR is the average over the packet reception ratios for all ideal neighbors of the source.

To define the idealized neighborhood distance in the simulation, the add delay of the three protocols under different SNR values are reported in Figure 18. As seen from

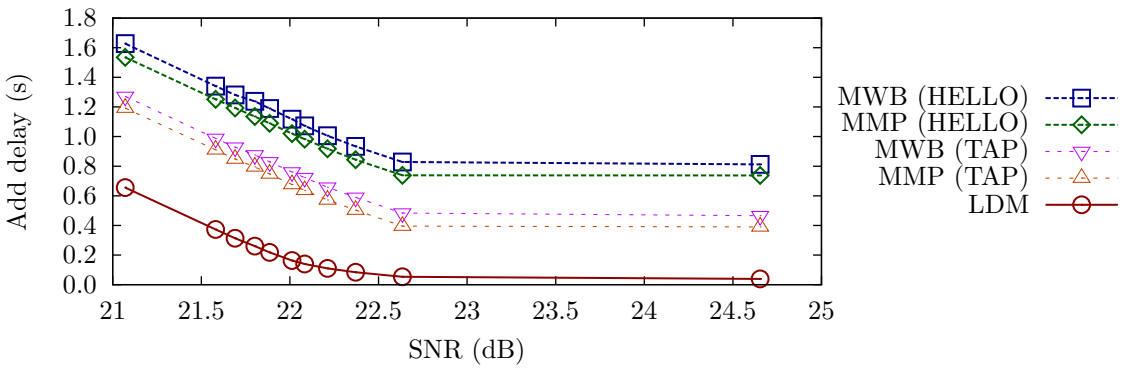


Figure 18: Add delay for each MAC layer multicast protocol under different SNR.

Figure 18, different ideal neighbor SNR yields different add delays. All subsequent simulation results are reported at the SNR value of 23.5 dB, which corresponds to an ideal neighbor distance of 41 meters under our simulation settings. Note that increasing or decreasing the chosen value by 1 dB has almost no impact on the add delay, meaning that the results are not very sensitive to this parameter value.

4.2.4 Speed of link discovery and its impact

In each simulation, forty nodes were placed randomly in the deployment area. Ten nodes were randomly selected as source nodes. One hop multicasts from the source nodes to nodes within their range were performed.

The add delays of different protocols are reported in Figure 19. Add delay was measured at the source nodes. We did not include non-source nodes in the evaluation since non-source nodes do not actively use their neighbor lists to transmit data. Therefore, a slight delay in maintaining a neighbor list does not affect the performance of those nodes. In a real network, where most nodes are active, either as original source nodes or as forwarding nodes, all active nodes act like source nodes from the MAC layer perspective.

The simulation results show that LDM is able to recognize a new neighbor that moves within its range significantly faster than other HELLO-based protocols. As can be seen from Figure 19, LDM has the shortest add delay among all three neighborhood

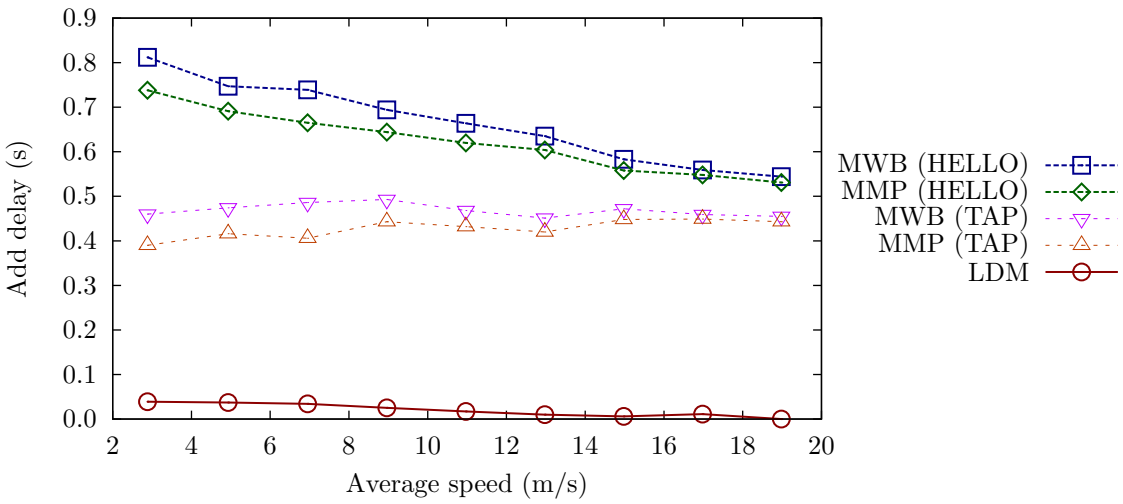


Figure 19: Average add delay of each protocol.

maintenance mechanisms. MMP and MWB have longer add delays due to the nature of the HELLO mechanism; a node is required to receive a HELLO message from a new neighbor to recognize it, which results in some delay since HELLO messages are sent periodically. In addition, a lost HELLO message will further delay recognition of a new neighbor, since HELLO messages are sent unreliably without retransmission.

The add delays of LDM are clustered closely around the mean whereas the add delays of MWB and MMP have higher variances. For instance, at the average speed of 2.86 m/s. LDM has an average add delay of 0.039 seconds with the standard deviation of 0.009 while MWB and MMP have average add delays of 0.812 seconds and 0.738 seconds and standard deviations of 0.31 and 0.27, respectively.

The multicast packet reception ratios of all the protocols are reported in Figure 20. As seen from the figure, LDM has the highest reliability among all the protocols. The reason for the difference between LDM and HELLO-based protocols is the ability of LDM to maintain better neighborhood information than the HELLO mechanism. MPRR decreased as average speed increased, as maintaining up-to-date neighborhood information is more difficult when nodes are moving at higher speeds. In the worst case, where an ideal neighbor relationship lasts only briefly, nodes using a HELLO

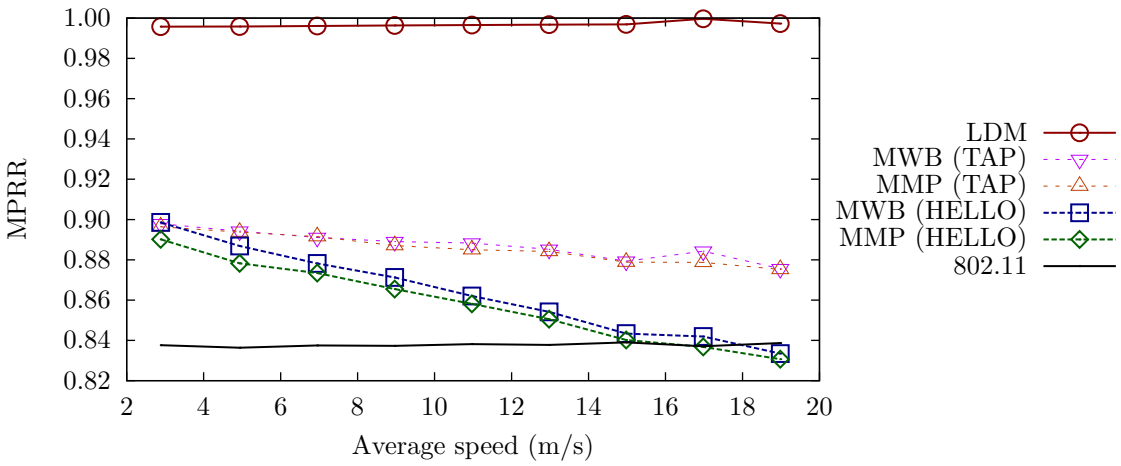


Figure 20: Multicast packet reception ratio for each protocol.

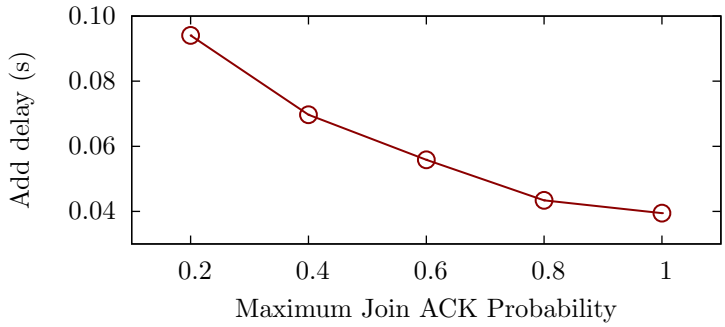


Figure 21: Add delay of LDM with different maximum Join ACK probabilities.

mechanism may not recognize the neighbor at all. TAP adjusts the HELLO message rate according to nodes' current speed to mitigate this problem; however, this also causes an increase in bandwidth consumption.

To study how the Join ACK probability affects add delay, the maximum Join ACK probability (P_{max}) was varied from 0.2 to 1.0. The mapping function from Join ACK level to Join ACK value was set to: $\Pr(Join) = \frac{1}{8} \cdot P_{max} \cdot (1 + L)$. The average add delays at an average speed of 2.86 m/s are reported in Figure 21.

The add delay of LDM increases as the maximum Join ACK probability decreases since the new node has lower probability to send a Join ACK to the source node. However, the average add delay of LDM is still significantly shorter than HELLO-based mechanisms.

4.2.5 Unidirectional and bidirectional links

In this section, we demonstrate that LDM has better ability to maintain neighborhood information than the HELLO mechanisms, particularly when links' states are frequently changing. In this simulation, 600 nodes were statically placed in the deployment area. Ten nodes were randomly selected as source nodes. A link between two nodes exists if they are separated by distance smaller than 41 m. Every link is either in a bidirectional state or a unidirectional state. The duration for the unidirectional state is uniformly distributed between 0 to 1 seconds. The duration for the bidirectional state is uniformly distributed between t_B to $t_B + 1$ seconds.

The total number of transmissions resulting from nodes that incorrectly classified neighbors as bidirectional are reported in Figure 22. Note that the y-axis is broken to better display the results. LDM has the lowest number of transmissions among all protocols. HELLO-based protocols have a higher number of false transmissions since nodes rely on HELLO messages and assume that links are bidirectional if a HELLO message is received. For instance, if a link between node A and node B is unidirectional from A to B, a HELLO message from A will be received by B. In this case, B will mistake A as a bidirectional neighbor. LDM does not assume that receiving a HELLO message from A indicates that the link is bidirectional and avoids this problem.

One possible approach to detect bidirectional neighbors in HELLO-based protocols is for the sender to include an incoming neighbor list in all HELLO messages. This approach was briefly mentioned in [71] although, to our knowledge, no existing implementations have adopted this technique. However, even if this technique is used, the delay in recognizing unidirectional links will be significantly higher than in our approach, because it could require several HELLO periods for the receiver on the unidirectional link to accurately determine that the sender of the link cannot receive its messages.

In the course of our experiments with the HELLO-based protocols, several other problems with their ability to distinguish incoming neighbors from bidirectional neighbors became apparent. First, in certain cases, the delay in recognizing a neighbor as bidirectional could be even higher than what was reported in Section 4.2.4. This problem arises in nodes that are not sending data (non-senders). The neighbor addition delay reported in Section 4.2.4 was for the multicast sender, which is actually a best case for HELLO-based protocols. When nodes do not send data, the only opportunity for other nodes to recognize them as incoming neighbors is through their HELLO messages. Thus, when a non-sender node first moves into the neighborhood of another node, if the other node sends its HELLO message before the non-sender node sends its HELLO message, the other node will not yet have recognized the non-sender as an incoming neighbor and so it will not include the non-sender node in its neighbor list. Thus, when the non-sender node receives the first HELLO message from the other node, it will not recognize the other node as a bidirectional neighbor. Only after the non-sender sends its HELLO message will the other node recognize the non-sender as an incoming neighbor. The other node will then include the non-sender in its neighbor list in its *second* HELLO message. Thus, the delay for the non-sender to recognize the other node as a bidirectional neighbor could be as high as two HELLO message periods.

A second problem arises when two sender nodes move within range of each other. At the point of first moving within range, neither node is included in the other node's neighbor set, so their reliable multicasts will not be sent to each other. Furthermore, since both nodes have frames to send, they no longer send out HELLO messages. Therefore, A only recognizes B as an incoming neighbor and B only recognizes A as an incoming neighbor (A and B hear the transmissions from each other). Since A and B only recognize each other as incoming neighbors, they will continue to not include each other in their multicasts.

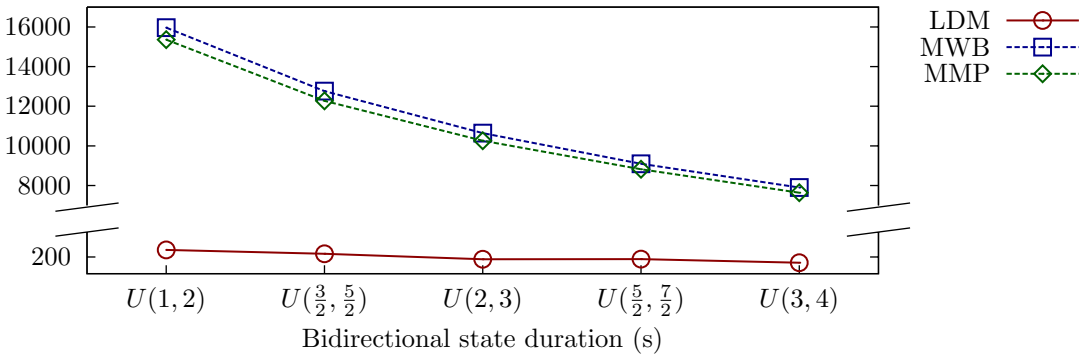


Figure 22: Total number of transmissions resulting from misclassification.

To try to address one or both of these problems, three possible options are

1. nodes can continuously send out HELLO messages

2. HELLO information can be piggybacked onto DATA frames, and

3. nodes can assume that all links are bidirectional.

Continuously sending out HELLO messages consumes more bandwidth. Piggybacking HELLO information onto DATA frames requires all nodes to operate in a promiscuous mode. Finally, assuming that all links are bidirectional will result in errors in neighbor classification [47, 72, 102]. LDM does not suffer from these problems since nodes can send join ACK to each other if the link is bidirectional. If the link is unidirectional, say from A to B, B will receive frames from A but A will not receive Join ACK from B. Thus, B will recognize A only as an incoming neighbor while A will not recognize B at all.

4.2.6 Effect of the application traffic model

The objective of the simulations reported in this section is to study the effect of different application traffic models on the performance of the LDM protocol. In this simulation, each source node has an on-off application. The duration of the on state is fixed to one second. The duration of the off state is uniformly distributed between

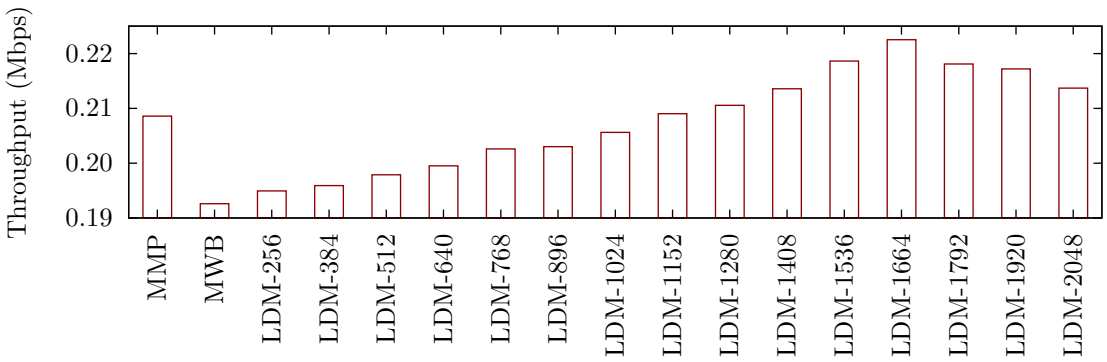


Figure 23: Average throughput for different MAC layer multicast protocols.

t to $t+1$ seconds ($U(t, t+1)$). For LDM protocol, a HELLO message is sent if a node has not send any DATA frame in the last one second.

The add delay of LDM increases from 0.039 s at the off state duration $U(1, 2)$ to 0.058 s at the off state duration $U(4.5, 5.5)$. Since LDM relies on an ACK from the new neighbor to recognize its presence, the new neighbor must be recognized through an explicit HELLO message during the off state. Thus, the add delay of LDM increases as the off state duration increases. However, the delay is still smaller than it would be if a traditional HELLO mechanism was used, since the new neighbor can send an ACK to the HELLO message, which allows the HELLO sender to recognize the new neighbor as a bidirectional neighbor as soon as the ACK is received.

4.2.7 Threshold-based transaction

In this simulation, we show that using frame size as a threshold for a four-way transaction results in better bandwidth utilization. In this simulation, 1000 nodes were statically placed in the deployment area. One hundred nodes were selected as source nodes. We varied the RTS threshold from 256 to 2048 bytes (LDM-*threshold*). The average throughput for each protocol is reported in Figure 23.

As seen from Figure 23, the RTS threshold in this case should be set to about 1664 bytes. Four-way transaction is useful when collision is likely. Setting the RTS

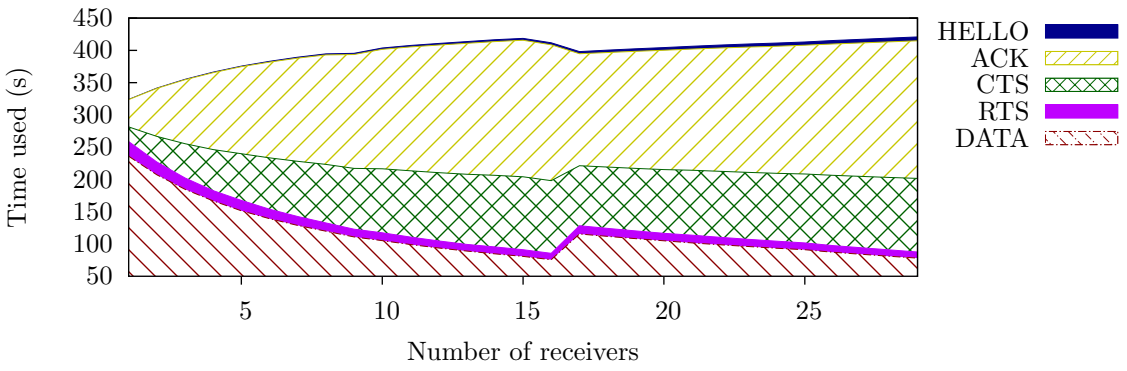


Figure 24: Transmission time used for different frame types.

threshold too low results in more transactions being protected by the RTS-CTS handshake than necessary. We note that selecting an appropriate RTS threshold depends on many factors, and thus should be left as a tunable parameter to be set by network administrators, which is the same practice recommended by the IEEE 802.11 standard for unicast frames.

4.2.8 Overhead of positive acknowledgement approach

Finally, we evaluate the overhead of a positive acknowledgement mechanism. A single source node is presented in this simulation. Varying number of receivers are placed within the transmission range of the source. The source node sends out packets at the rate of 30 Mbps. This simulation represents the *best*-case scenario where all receivers are within the transmission range of the source, and no contending application. The total time used to transmit different frames are reported as stacked areas in Figure 24.

As seen from Figure 24, as the number of receivers increases, the total time used for transmitting control frames also increases. The total time used to transmit control frames exceeds the total time used to transmit DATA frames when more than four receivers are present. The increase in DATA transmission from 17 receivers to 18 receivers was due to the number of receivers' addresses is too large to be fitted into one transaction. In this case, the sender had to split the multicast transaction into two subtransactions.

If a very large number of receivers are present and the multicast involves an access point and multiple receivers, an approach like the 802.11aa Block ACK can be used to reduce the overhead of the control messages [9].

4.3 Discussion

In this chapter, we have proposed an extension to the IEEE 802.11 framework. The proposed protocol's goals are to provide reliability for a multicast frame, and to enable neighborhood maintenance at the MAC layer. Simulation results show that our proposed protocol is able to achieve higher reliability than other reliable multicast protocol. Our proposed protocol is able to quickly recognize a new node moving within its range, while reducing the use of HELLO messages in the network. Our protocol is also able to distinguish between incoming neighbors and bidirectional neighbors without high overhead.

It is clear that providing reliability for multicast operation at the MAC layer has the potential to enhance the performance of network layer multicast routing. However, one of the problems of network layer multicast routing is that the children nodes are likely to forward the multicast frames at the same time, resulting in higher interference experienced by their children. Figure 25 illustrates this scenario. In Figure 25, a node at level $d-1$ multicasts frame to its children nodes at level d. Both nodes at level d then forward the frames at almost the same time, resulting in a high level of interference at nodes at level $d+1$. If the interference is high enough, the transmission from the nodes at level d to the nodes at level $d+1$ might not be successful and require retransmission.

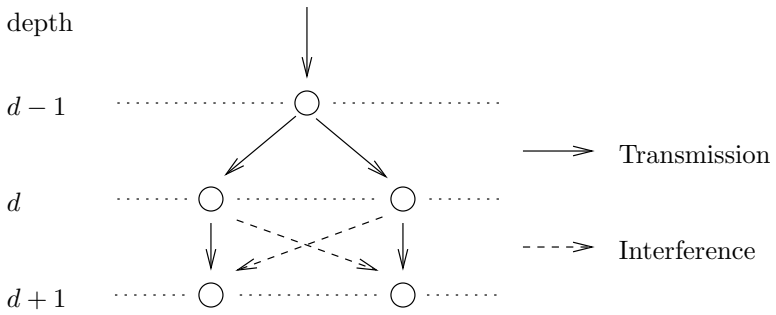


Figure 25: Synchronization of frames forwarding after a multicast tree branch.

CHAPTER V

INTERFERENCE-AWARE MULTICAST FOR WIRELESS MULTIHOP NETWORKS

In multicast, a single message is delivered to a group of destinations in a network. A major limitation of research in this area, to date, is that the vast majority of works ignore interference, which is a significant factor in wireless multihop networks. The few works that do consider interference use inaccurate models. In this chapter, we carry out the first research study of end-to-end multicast routing structures for wireless multihop networks that accounts for interference using accurate interference models. We design new interference-aware multicast routing structures and show that their performance substantially exceeds that of existing multicast algorithms that do not account for interference.

Most tree-based protocols have been based on shortest path trees or Steiner trees. The goal of shortest path trees (e.g. [43, 70]) is to minimize the distance between the source and each destination, while the goal of Steiner trees (e.g. [35, 62]) is to minimize the sum of the distances in the multicast tree. A few studies comparing these predominant tree structures have been done. Ruiz and Gomez-Skarmeta [77] studied shortest path trees and Steiner trees. The authors argued that Steiner tree is not appropriate for wireless networks and proposed that the problem should be reformulated to minimize the cost in terms of the number of forwarding nodes. They proposed a greedy heuristic algorithm, called MNT, and showed that the proposed algorithm is able to reduce the number of forwarding nodes. Nguyen [64] revisited the study and evaluated the performance of shortest path trees, Steiner trees, and the MNT algorithm in terms of packet delivery ratio. The author showed that shortest

path trees offer the best performance in terms of packet delivery ratio. However, neither of these studies accounted for interference in their evaluations.

Other work that studied multicast scaling law and structure are [56, 75, 79]. The authors studied the asymptotic multicast capacity of multihop wireless networks. In [79], a comb structure for multicast trees that achieves capacity in the order sense was proposed. While the work did account for interference, they used the protocol model for interference, which is not as accurate as the physical interference model, and they were concerned primarily with asymptotic scaling results, rather than best performance on finite networks.

Scheduling is an important aspect of wireless multihop networks with interference. Scheduling can increase network throughput by letting devices access the channel in an orderly fashion instead of the more conservative contention-based access. The only multicast scheduling works of which we are aware are [27, 90]. However, [90] is primarily concerned with power control and scheduling plays only a minor role, while [27] only deals with one-hop (not end-to-end) multicast.

In this chapter, we consider the problem of interference-aware multicast routing tree in wireless multihop networks, using an accurate physical interference model. First, we study the problem for low-intensity multicast. We classify nodes into different classes and derive optimal interference-aware routing strategies for each class. Based on the analyses, we propose a new multicast routing structure based on Steiner trees for the low-intensity case. Next, we consider a general multicast scenario and propose a second new multicast routing structure that is not based on Steiner tree. We evaluate our proposed structures through simulation in both the TDMA and CSMA/CA settings. Simulation results demonstrate that, compared to existing approaches, our proposed structures reduce the schedule length up to 57% in TDMA networks and achieve up to 41% higher goodput in CSMA/CA networks.

5.1 System model and problem formulation

We consider a communication graph $G = (V, E)$, where V is a set of all wireless nodes. An edge $(u, v) \in E$ if and only if $d(u, v) \leq r_t$, where $d(u, v)$ is the Euclidean distance between nodes u and v and r_t is the maximum transmission range. We are given a multicast source $s \in V$ and a set of multicast destinations $M \subset V$. The problem is to construct a multicast tree rooted at s that spans M, along with a partition of a set of non-leaf nodes in the multicast tree $S_1, S_2, \ldots, S_k$, where S_i is a set of feasible transmissions with the given interference model. A feasible transmission set is a set where, if all nodes in the set transmit to their respective receivers at the same time, all receivers will successfully receive the transmissions. We call the partition $S_1, S_2, \ldots, S_k$ a schedule with schedule length k. Our goal is to construct a minimum length schedule.

We adopt the classical model for radio signal propagation, which is referred to as the log-distance propagation loss model. The radio signal strength at a distance d from the transmitter is given by $\frac{P_t}{d^\alpha}$, where P_t is the transmission power and α is the path loss coefficient. We assume that nodes are not equipped with interference cancellation capabilities.

We consider the physical interference (PI) model [36]. In the PI model, interference from all concurrent transmitters in the network, no matter how distant, is factored into the signal-to-interference ratio (SIR) value at the receiver. Specifically, the transmission will be correctly received by the receiver if and only if the SIR value at the receiver is larger than the SIR threshold ($\mathrm{SIR}_{\min}$).

5.2 Interference-aware routing for low intensity multicast

In this section, we consider low intensity multicast, which is defined as a situation where the time between two consecutive packets generated by the source is greater than the time it takes to deliver one multicast packet to all destinations. We first

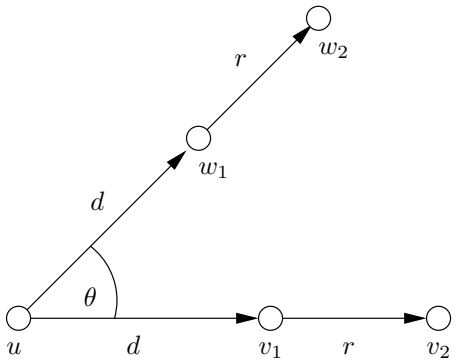


Figure 26: A branching node u branching into two paths.

consider an ideal network where we are given a source node s and a set of destination nodes M and we are able to place extra nodes anywhere in the network when building a multicast tree. Our goal is to construct a tree rooted at s that spans M while taking interference into account. To achieve this, we classify nodes in a multicast tree into three classes.

1. Leaf nodes – nodes with no child.

2. Path nodes – nodes with exactly one child.

3. Branching nodes – nodes with more than one child.

According to our classification, leaf nodes do not forward data in the multicast tree and so they do not generate interference. Furthermore, along a single path, there is no interference between nodes since there is only one packet being transmitted at a time. Thus, the optimal structure along a single path is for each transmission to travel as far as possible, i.e. to separate consecutive nodes by the transmission range. Next, we determine optimal structures involving branching nodes, which are non-trivial to analyze.

5.2.1 Branching nodes with two children

First, consider a branching node u with two children v_1 and w_1 as shown in Figure 26. The distance of the first hop from u to v_1 and u to w_1 is d, where $d \leq r_t$. The first

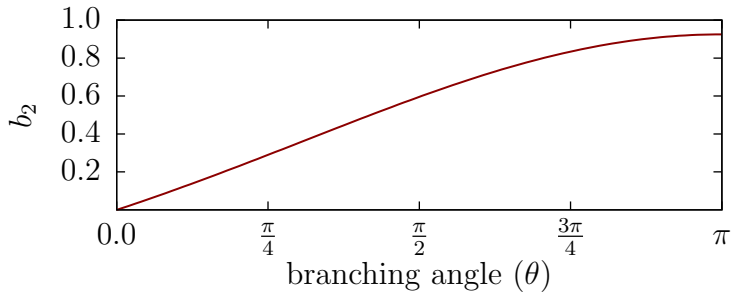


Figure 27: Values of b_2 with different branching angle θ.

transmission from u to v and w is done by using a multicast or broadcast. The second transmissions from v_1 to v_2 and from w_1 to w_2 occur at almost the same time. The cross interference from the concurrent transmission is given by

$$\frac{P_t}{[r^2 + 2d(1-\cos\theta)r + 2d^2(1-\cos\theta)]^{\alpha/2}}.$$

Combining the received signal strength and the interference, we set SIR to SIR_{min} and convert to decibel to get

$$0 = \left(1 - 10^{\frac{\text{SIR}^{\text{dB}}_{\text{min}}}{5\alpha}}\right) r^2 + 2d\,(1-\cos\theta)\,r + 2\,(1-\cos\theta)\,d^2.$$

Solving the equation, we get $r = b_2 \cdot d$ where

$$b_2 = \frac{1 - \cos\theta + \sqrt{(\cos\theta - 1)(1 - 2\cdot 10^{(\frac{\text{SIR}^{\text{dB}}_{\text{min}}}{5\alpha})} + \cos\theta)}}{-1 + 10^{(\frac{\text{SIR}^{\text{dB}}_{\text{min}}}{5\alpha})}}.$$

The result shows that the distance between nodes after the branching point is proportional to the distance of the first hop. We show the values of b_2 at different branching angle θ in Figure 27.

Let i be the depth from u and r_i be a distance between a node at depth i and a node at depth $i+1$. For all $i > 0$, we can consider d as a sum of all r_j, where $0 \leq j < i$, we get

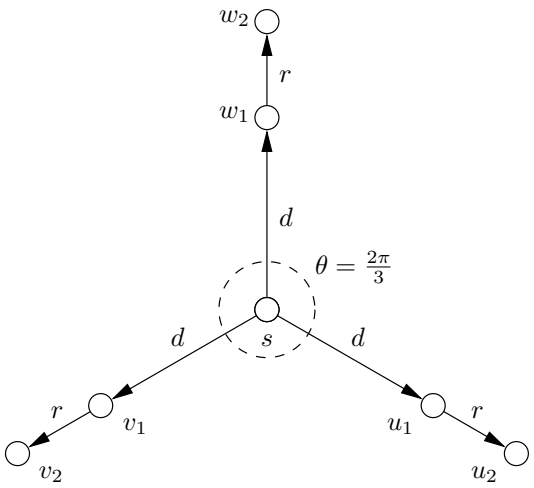


Figure 28: A branching node s branching into three paths.

$$r_i = b_2 \sum_{j=0}^{i-1} r_j = b_2(b_2+1)^{i-1} r_0, \text{where } r_i \leq r_t \text{ and } r_0 \leq r_t$$

This shows that the distance between v_i and v_{i+1} grows as v_i gets further away from u until the distance reaches r_t, which is the transmission range and corresponds to the optimal distance for the single-path case.

5.2.2 Branching nodes with three children

Next, we consider a branching node s with three children u_1, v_1, and w_1 as shown in Figure 28. Applying a similar analysis to the two-child case, we get $r = b_3 \cdot d$ where

$$b_3 = \frac{3 + \sqrt{9 - 12(1 - 10^{\frac{10\log 2 + \mathrm{SIR}_{\min}^{\mathrm{dB}}}{5\alpha}})}}{2(10^{\frac{10\log 2 + \mathrm{SIR}_{\min}^{\mathrm{dB}}}{5\alpha}} - 1)}.$$

Again, the distance between nodes grows as nodes get further away from the branching node, albeit in a slightly different manner. Here also, the distance will eventually reach the limit r_t.

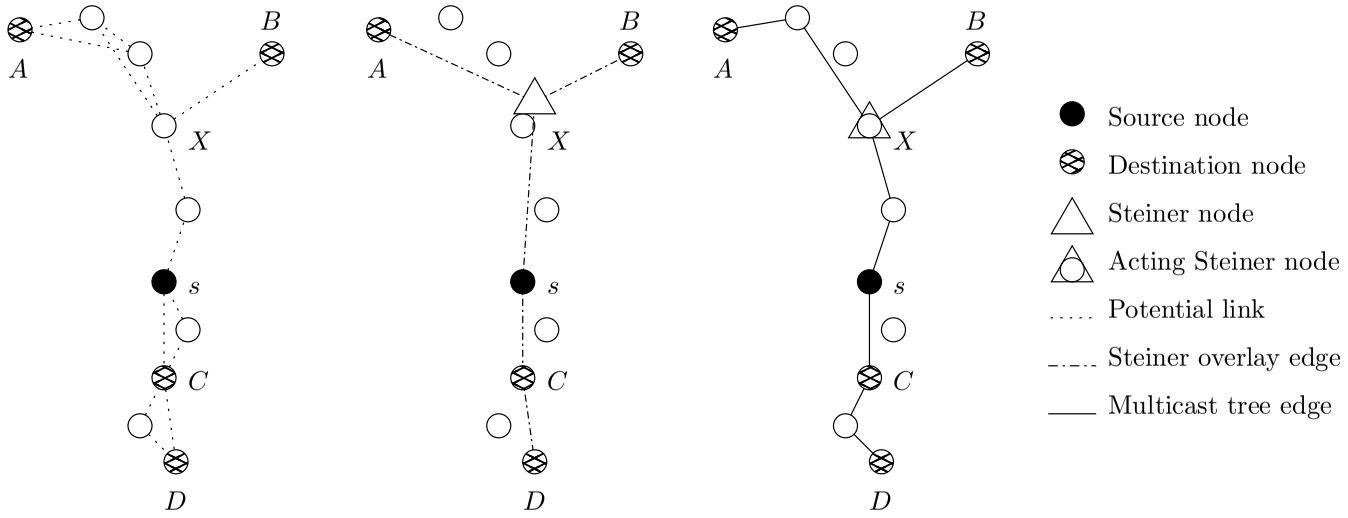
Using the preceding analyses, we present an approximation algorithm to build a multicast tree. The algorithm applies different routing strategies depending on which is the applicable case (single path, two-way branch, or three-way branch).

5.2.3 Interference-aware Steiner tree algorithm (IAST)

Since the optimal routing strategies are different for different classes, our goal is to use different routing strategies for different classes when building a multicast tree. However, one of the difficulties of applying this approach is identifying where branching should take place. Our goal is to first identify branching nodes locations and use different routing strategies for different classes when building a multicast tree.

The high level idea of the interference-aware Steiner tree routing algorithm is as follows: we are given nodes that must be connected in a multicast tree. These nodes are the source node and all the destination nodes. The first step is to identify how these nodes should be connected in a tree. The algorithm uses a Euclidean Steiner Tree approximation algorithm to locate ideal Steiner nodes in the network, using $M \cup \{s\}$ as input. We call the returned Euclidean Steiner tree a Steiner overlay tree since it shows the "big picture" connections between nodes in the network. An edge between two nodes in the Steiner overlay tree suggests that the two nodes should be connected by a path in the original graph. Note that we can view $M \cup \{s\}$ as a multicast group. The multicast group will have one Steiner overlay tree regardless of which node is a source node.

For each Steiner node, the algorithm finds a real node nearest to the ideal Steiner node location to act as the Steiner node. The algorithm consults the Steiner overlay tree to determine if the current node should be considered as the source node, a path node, or a branching node. The algorithm uses different distances for different node classes based on the analyses in Section 5.2.1 and Section 5.2.2. Note that Steiner trees contain only two-way branches and up to one three-way branch (at the source), meaning that our analyses are sufficient to handle all cases. Figure 29 shows an example of steps to build a multicast tree of IAST.



(a) network topology (b) a Steiner overlay tree (c) a final multicast tree

Figure 29: An example of building a multicast tree by IAST algorithm.

5.3 Interference-aware routing for general multicast

In this section, we consider a more general multicast scenario where the source node may begin transmission of the next packet before all receivers have received the previous packets. As a result, there may be more than one application layer packet being forwarded in the network. Multiple application layer packets being forwarded in the network means that interference in the network will be higher than the low intensity multicast case.

Since more than one application layer packet may be presented in the network, we are unable to directly apply the analyses from Section 5.2. The extra interference means that the distances between nodes must be shortened to allow for concurrent transmissions. To tackle this problem, we introduce a scaling factor, f, to be used to scale down the distances analyzed from the Section 5.2. Our goal is to find the most appropriate value for the scaling factor f.

5.3.1 Scaling factor in path nodes

Consider a linear path in one dimension with infinite length as shown in Figure 30. If all nodes are equally separated by a separation $r = r_t$, then all links in the path are on the edge of the SINR threshold and no concurrent transmission is possible. The schedule length in this case will be on the order of the length of the path. This phenomenon is known as the black-gray link paradox [12]. Edges with distance exactly equal to the transmission range are referred to as "black" links, and they do not allow a single concurrent transmission, no matter how distant. Edges with distance slightly below the transmission range are referred to as "gray" links, and they do allow concurrent transmission, although the allowable spatial separation might be quite large.

Given the black-gray link paradox, if the separations between nodes are scaled down to $r = f \cdot r_t$, where $0 < f < 1$, it should be possible to have multiple nodes

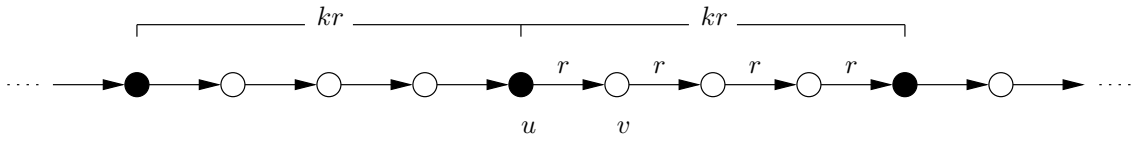


Figure 30: An infinitely-long, equally-spaced linear network.

transmit concurrently. Suppose that the schedule length is k. Two consecutive transmitters are separated by a distance $k \cdot r$. Consider a transmission from node u to node v, the total interference experienced by v is given by

$$\sum_{i=1}^{\infty} \left[\frac{P_t}{(ikr + r)^{\alpha}} + \frac{P_t}{(ikr - r)^{\alpha}} \right],$$

The SIR at the receiver v can be obtained by combining the received signal strength and the total interference at v, the SIR at v is given by the equation

$$\mathrm{SIR}(v) = \frac{1}{\sum_{i=1}^{\infty} \left[\frac{1}{(ik+1)^{\alpha}} + \frac{1}{(ik-1)^{\alpha}} \right]}.$$

The transmission will be correctly received by v if and only if $\mathrm{SIR}(v) \geq \mathrm{SIR}_{\min}$; we get the following inequality:[1]

$$\sum_{i=1}^{\infty} \left[\frac{1}{(ik + 1)^{\alpha}} + \frac{1}{(ik - 1)^{\alpha}} \right] \leq \mathrm{SIR}_{\min}^{-1}. \tag{1}$$

We use the convergence integral test to the left hand side of (1) and get

$$\frac{1}{(k + 1)^{\alpha}} + \frac{1}{(k - 1)^{\alpha}} + \frac{(k + 1)^{1-\alpha} + (k - 1)^{1-\alpha}}{k(\alpha - 1)} \leq \mathrm{SIR}_{\min}^{-1}. \tag{2}$$

Equation (2) shows that according to SIR, the schedule length of an infinitely-long equally-spaced linear network depends only on the path-loss coefficient and the $\mathrm{SIR}_{\min}$. Thus, the scaling factor can increase spatial reuse but Equation (2) shows that the schedule length will eventually converge.

For the IAST algorithm, the goal of the scaling factor is to accommodate concurrent transmissions from other nodes. We scale down the distance used when building

[1] We assume that $\mathrm{SIR}_{\min} > 0$, where $\mathrm{SIR}_{\min}$ is in a linear ratio.

a multicast tree by a factor f instead of using distances directly from the analyses in Section 5.2.

Since the application of the scaling factor is not limited to IAST, we propose our second interference-aware algorithm.

5.3.2 Fixed-distance tree merging algorithm

The IAST algorithm requires global knowledge of the network to approximate a Euclidean Steiner tree and identify candidate nodes to act as Steiner nodes. Our second algorithm, the fixed-distance tree merging (FTM) algorithm, does not require global knowledge of the network. The algorithm grows the source tree by merging the source tree with the nearest destination node until all destination nodes are connected.

The high level description of FTM algorithm is as follow. The algorithm takes a preferred scaling factor (f_p) as an input, where $0 < f_p \leq 1$. The algorithm uses the same distance $r = f_p \cdot r_t$ when building a multicast tree. For each destination node, the algorithm finds a shortest path in terms of hop counts from the receiver to all nodes in the network. The algorithm now knows the destination node nearest to the source tree.

The algorithm grows the tree from the source tree towards the destination node by using the following criteria when selecting the next hop node.

1. the next hop node *must* be closer to the destination node
2. if there exist multiple next hop nodes, the algorithm picks the next hop node that is closest to all other remaining destination nodes

After the selected destination node is merged with the source tree, the algorithm selects the next destination node nearest to the new source tree and grows the tree toward the selected node until all destination nodes are included in the tree. In the case where node density is too low and growing the tree with $r = f_p \cdot r_t$ is not possible, FTM gradually increases r until the selected node is successfully merged or $r = r_t$.

5.3.3 Scheduling algorithm

To accommodate the routing tree with different distances for different node classes, we propose a modified version of GreedyPhysical scheduling algorithm [13], called Tree-Based GreedyPhysical. The idea of Tree-Based GreedyPhysical is to schedule all forwarding nodes at the same depth in the same slot until no more forwarding nodes can be accommodated in the slot. Tree-Based GreedyPhysical then reverts back to the original GreedyPhysical.

5.4 Performance evaluation

We evaluate our algorithms in four different settings. First, we start by evaluating our algorithms in a low intensity multicast setting where the analyses in Section 5.2 can be applied directly. Next, we evaluate our algorithms in a general multicast scenario where the analyses cannot be applied directly and scheduling is required. We also evaluate our algorithms in a CSMA/CA setting where nodes access the channel in a contention-based manner rather than with TDMA. Finally, we evaluate our algorithms in a CSMA/CA setting where wireless links are bursty.

5.4.1 Simulation parameters and assumptions

We use ns-3.15 simulator to evaluate all algorithms. We use a physical model of 802.11g at the data rate of 6 Mbps in the simulation. All nodes use the transmission power of 40 mW and thermal noise is computed at 290K. In all simulations, 2000 nodes were uniformly distributed in a deployment area of 1000 m by 1000 m.[2] Common simulation parameters in Table 2 are used in all simulations, unless stated otherwise.

We compare IAST and FTM against three existing tree-based structures: MNT [77], SPT [64], and COMB [79]. For IAST algorithm, we use GeoSteiner [2, 91] to find a Euclidean Steiner Tree. All results reported are averaged from 1000 simulations.

[2]With 802.11g at 6 Mbps, transmission range is rather small and fairly high node density is required for the network to be connected.

Table 2: Simulation parameters used to evaluate multicast routing structures.

Parameter	Value
Deployment area	1000 m by 1000 m
Number of nodes	2000
Mobility model	constant position
Propagation loss model	log-distance
Path-loss exponent	3
Transmission power	40 mW
Thermal noise	290K
Device	IEEE 802.11g
Default transmission mode	6 Mbps

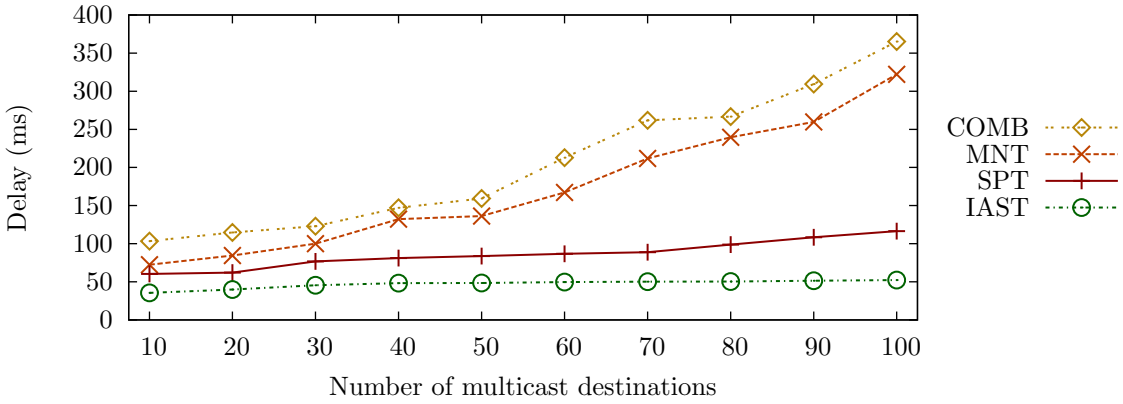


Figure 31: End-to-end delivery delay of different multicast routing structures.

5.4.2 Low intensity multicast

We begin our evaluation with a low intensity multicast scenario. We varied the number of multicast destination nodes from 10 to 100. To prevent multiple nodes from forwarding packets at exactly the same time, we inserted random delay between 0 μs and 1000 μs before nodes forward packets to their children. We measured the delay between the time when the source node transmits the packet and the time when all destinations have received the packets. The simulation results are reported in Figure 31.

As seen from Figure 31, the IAST algorithm has the lowest delivery delay among all algorithms, even though SPT has the shortest distance between the source and every destination. SPT has a larger number of nodes and this creates more contention

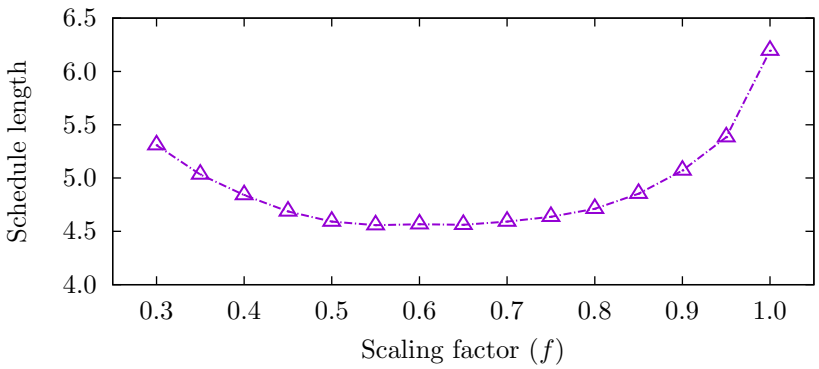


Figure 32: IAST schedule length with different scaling factors.

in the network, which means that nodes have to delay their transmissions when the channel is sensed as busy. MNT and COMB both have even longer delays, because they have both longer paths and higher contention.

5.4.3 General multicast

In this section, we consider a general multicast application where the application generates packets faster than the time it takes to deliver packets to all destinations. Thus, we use a scaling factor and scheduling in the algorithms.

We first evaluate the scaling factor since it is a significant parameter affecting IAST and FTM performances. In this simulation, the number of multicast destinations is fixed at 10. We vary the scaling factor from 0.3 to 1.0 and collect the schedule lengths returned by IAST algorithm. We did not use the scaling factor below 0.3 since the network became disconnected in some simulations. The simulation results are reported in Figure 32.

Figure 32 confirms that using different scaling factor affects IAST performance. If the scaling factor is too small, the extra nodes included in the multicast tree outweigh the gain of spatial reuse. If the scaling factor is too large, spatial reuse is not possible. However, performance is quite stable across a wide range of scaling factors, e.g. 0.5 to 0.7. Based on this result, we have set the scaling factor to 0.7 in the remaining simulations.

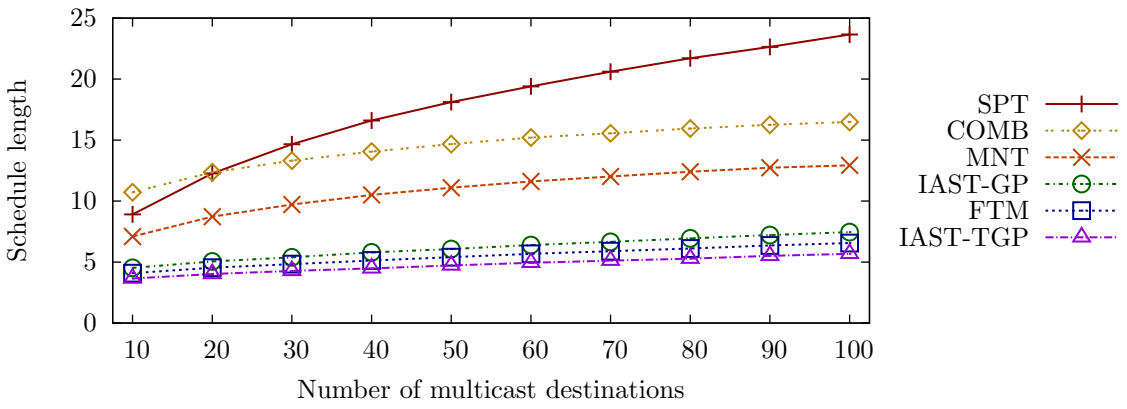


Figure 33: Schedule length with varying number of destinations.

We now consider the schedule length produced by different multicast routing structures, including our IAST and FTM structures. For these simulations, we varied the number of destinations from 10 to 100. There are multiple possible combinations between routing and scheduling algorithms. IAST-GP refers to IAST routing combined with the GreedyPhysical scheduling, IAST-TGP refers to IAST routing combined with Tree-Based GreedyPhysical. For MNT, SPT and COMB, we report the results from GreedyPhysical only since the difference between GreedyPhysical and Tree-Based GreedyPhysical are not statistically significant with 1000 simulations. The schedule lengths are reported in Figure 33.

The results of Figure 33 show that the schedule lengths of IAST and FTM are substantially shorter than SPT, MNT, and COMB. FTM schedules are approximately half the length of MNT schedules, 1/3 the length of COMB schedules, and less than 1/4 the length of SPT schedules for higher numbers of destinations. The shorter schedule lengths means that IAST (or FTM) can support more transmissions than other structures within the same time period. As expected, SPT and MNT achieved their respective goals of shortest average path length and minimum number of forwarding nodes. However, multicast routing structures built by MNT and SPT mostly consist of black links that cannot be activated concurrently with any others. As a

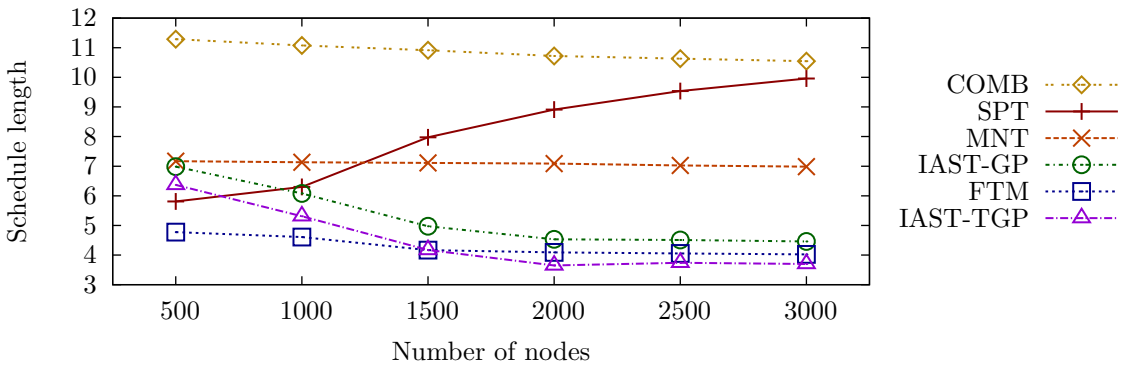


Figure 34: Schedule length of different algorithms with varying node density.

result, most of the forwarding nodes selected by MNT and SPT must be scheduled sequentially, which increases the overall schedule length. The presence of black links is even more problematic in SPT since the number of forwarding nodes is not taken into account, resulting in a very large number of forwarding nodes that must be scheduled sequentially. For COMB structure, the rigid structure of COMB results in longer schedule length.

We also evaluate the performance of all algorithms with varying network density. We kept the number of destinations at 10 and varied the number of nodes in the network from 500 to 3000. The minimum number of nodes could not drop below 500 nodes since the network becomes disconnected in some simulations with our settings of 802.11g at 6Mbps. The schedule lengths are reported in Figure 34.

As seen from Figure 34, the schedule lengths of IAST and FTM algorithms increase as the node density decreases. This effect is particularly noticeable for IAST, because the closest nodes to ideal Steiner node locations can be quite far, meaning that the tree structures begin to deviate significantly from ideal Steiner trees. Since FTM is not built on Steiner trees, it is less susceptible to the lack of ideal node locations and its schedule length does not increase as dramatically for lower node densities. Even at the lowest node density, FTM's schedule length is about 17% shorter than SPT's, almost 34% lower than MNT's, and more than 55% lower than COMB's.

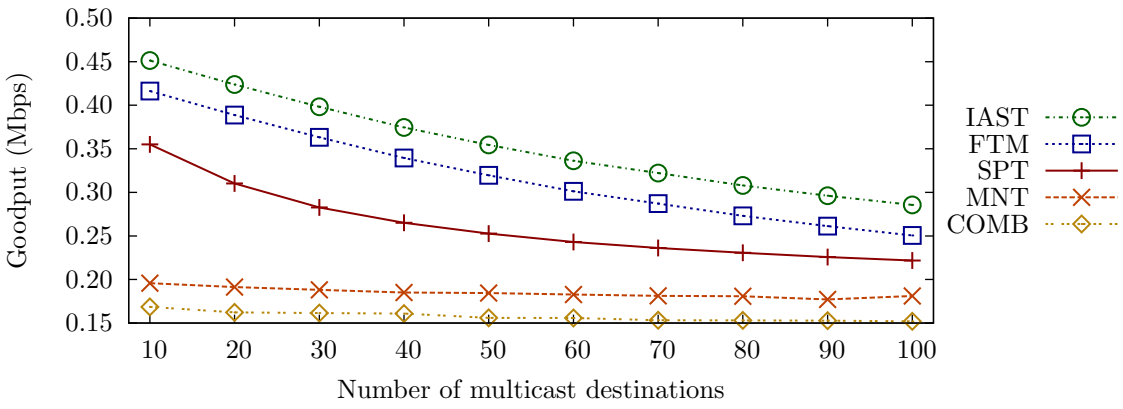


Figure 35: CSMA goodput of different multicast routing structures.

5.4.4 Contention-based channel access

Although in CSMA/CA networks, transmissions cannot be precisely scheduled, we hypothesized that the greater transmission concurrency facilitated by our interference-aware algorithms would still translate into better performance for the CSMA/CA case. To evaluate this, we performed ns-3 simulations using the standard ns-3 802.11 model. To prevent multiple nodes from forwarding packets at exactly the same time, we inserted random delay before nodes forward packets to their children. The random delay was chosen uniformly between 0 μs and 1000 μs. We measured the maximum goodput and report the results in Figure 35. Note that there is no scheduling in this CSMA/CA simulation, thus, only one variation of IAST is reported.

As seen from Figure 35, IAST and FTM still outperform SPT and MNT even without explicit scheduling. In IAST and FTM, concurrent transmission is possible while black links in SPT and MNT are intolerant of even a small interference from concurrent transmissions. The random delay helps mitigate the problem in SPT and MNT. Without the random delay, SPT and MNT performance was even lower than the reported results. Thus, our interference-aware multicast routing structures permit higher levels of concurrency and this results in significant performance benefits even without explicit scheduling.

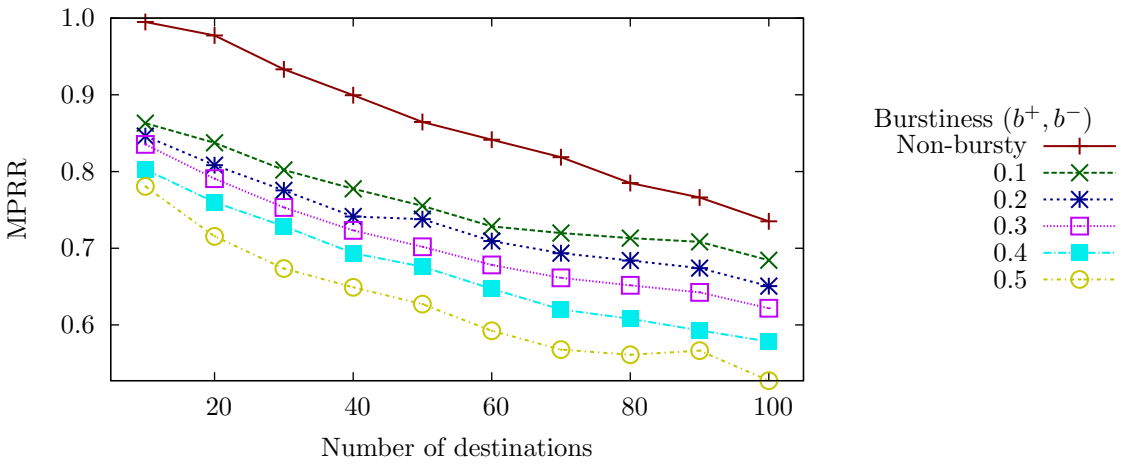


Figure 36: MPRR of the IAST protocol with varying wireless link burstiness.

5.4.5 Bursty wireless channels

One of the advantages of the mesh-based multicast routing protocols over tree-based routing protocols is that multicast packets have more than one possible path to reach the destination. In tree-based multicast, multicast packets have only one possible path to reach the destination. Any failed transmission along the path will result in the destinations missing the multicast packet.

In this section, we evaluate the performance of our interference-aware multicast protocols when the wireless channel is bursty. When the wireless channel is bursty, transmission between two nodes have higher probability to fail than when the wireless channel is not bursty. As a result, wireless link burstiness can adversely affect the performance of our interference-aware multicast protocols since the protocols rely on tree structures to deliver multicast packets.

To evaluate the performance of our protocols, we performed ns-3 simulations using the standard ns-3 802.11 model. We used the ideal bursty link BPA function (see Section 3.2.2) with varying burstiness to simulate bursty links. We measured the multicast packet reception ratio and report the results in Figure 36.

As seen from Figure 36, the packet reception ratio decreases as link burstiness increases. This result is expected since transmission between nodes are more prone to error when link burstiness increases. Since the IAST protocol uses tree as the underlying routing structure, transmission failure will lower the multicast packet delivery rate at the destinations.

5.5 Discussion

In this chapter, we have studied the problem of multicast routing and scheduling for wireless multihop networks. We have classified nodes in multicast trees into different classes and shown by analyzing an accurate physical interference model that different classes require different routing strategies to produce optimal schedules. Based on these analyses, we have proposed two joint routing and scheduling algorithms for multicasting in wireless multihop networks. We have evaluated the performance of different algorithms through simulation and shown that the proposed algorithms outperform other algorithms in terms of schedule lengths and goodput. Finally, we have studied the performance of our interference-aware Steiner multicast tree protocol when the wireless channel is bursty.

CHAPTER VI

INTERFERENCE-AWARE MESH MULTICAST FOR WIRELESS MULTIHOP NETWORKS

One of the fundamental properties of a tree is that, for each pair of node u and v in a tree, there exists exactly one u, v path. This property means that a single transmission failure or a single node failure will disconnect the tree. When the graph is disconnected, the multicast destination has no other path to receive the multicast packet, which results in lower multicast packet reception ratio.

One possible solution to mitigate this problem is to use a mesh as the underlying routing structure instead of a tree. In general, a mesh is a connected graph where there is more than one path from a multicast source to each multicast destination. These extra paths can deliver the multicast packets to the destinations if the transmissions on other paths have failed.

Multiple multicast routing protocols use mesh as their underlying multicast routing structures. Examples of mesh-based multicast routing protocols are ODMRP [52] and PUMA [88]. However, most of the mesh-based multicast protocols are concerned with the problem of building and maintaining a multicast mesh structure efficiently without taking interference into account.

A few studies have been done on the theoretical aspect of the multicast mesh structures. Zhao and others [101] proposed four heuristics to build a resilient multicast mesh structure. Two heuristics, NDT and RNDT, build a multicast mesh by merging two node-disjoint MNT multicast trees [77] to form a multicast mesh. The other two heuristics, SDM and MDM, build a multicast mesh by finding a pair of node-disjoint shortest paths from the source node to each multicast receiver, then merging all the

node-disjoint paths to form a multicast mesh. The proposed heuristics, however, did not take interference into account.

In this chapter, we propose two algorithms to extend the interference-aware multicast tree to form an interference-aware multicast mesh. The first algorithm creates an overlay mesh network from the overlay multicast tree by placing two extra nodes for each overlay link on the overlay multicast tree. The second algorithm uses Delaunay triangulation to identify the positions of the extra nodes. We evaluate both interference-aware mesh structures against the interference-aware multicast tree and other mesh structures through simulation. We design the simulation with the goal to evaluate the performance of multicast routing structures in two possible causes of graph disconnection – link failure and node failure. We show that, in both cases, our proposed interference-aware mesh structures outperform the interference-aware multicast tree and other mesh structures that do not take interference into consideration.

6.1 Overlay link extension algorithm

We propose our first tree extension algorithm, called overlay link extension algorithm (OLE). The idea of our first algorithm to extend a multicast tree to form a mesh is to place two extra overlay nodes for each overlay link in the overlay multicast tree. We call the extra nodes shadow nodes. The goal is to create two redundant paths for every overlay link in the multicast tree. Assuming that the distance between the two overlay nodes is l, we place each shadow node such that the distance between the shadow node and each of the overlay node is also l. The general idea of our first algorithm is depicted in Figure 37.

By introducing two redundant paths between the two overlay nodes, we have created one branching point and one merging point where the transmissions converge to a single overlay node. Next, we determine optimal structures involving the branching point and the merging point.

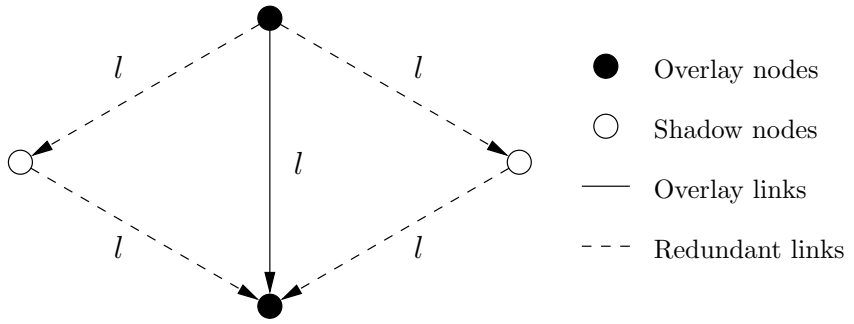
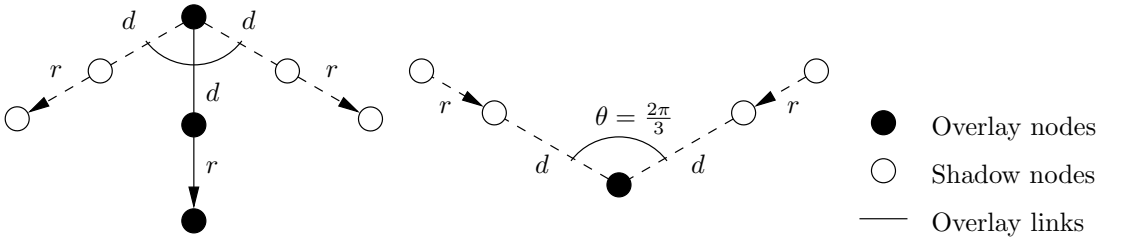


Figure 37: The underlying idea of the overlay link extension.



(a) a branching point (b) a merging point

Figure 38: Two scenarios to be considered of the overlay link extension algorithm.

6.1.1 Mesh branching nodes

Consider a mesh branching node that branches into three paths – one overlay path and two redundant paths as shown in Figure 38 (a). The receiver on the overlay path is the node that experiences the largest interference among the three receivers. For simplicity, we assume that the distances between nodes at the same depth from the mesh branching node are identical. The total interference at the receiver on the overlay path is given by

$$\frac{2P_t}{(d^2 + dr + r^2)^{\alpha/2}}.$$

Combining the received signal strength and the interference, we set SIR to $\mathrm{SIR}_{\min}$ and convert to decibel to get

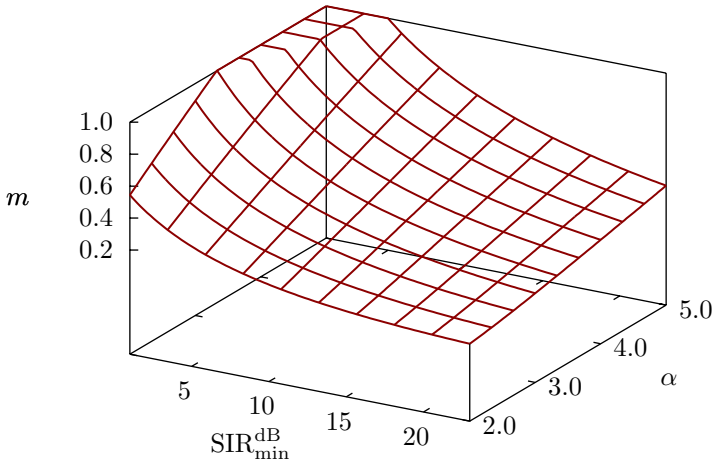


Figure 39: Values of m with different $\mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}$ and α.

$$0 = (1 - 10^{\frac{10\log 2 + \mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}}{5\alpha}})r^2 + dr + d^2$$

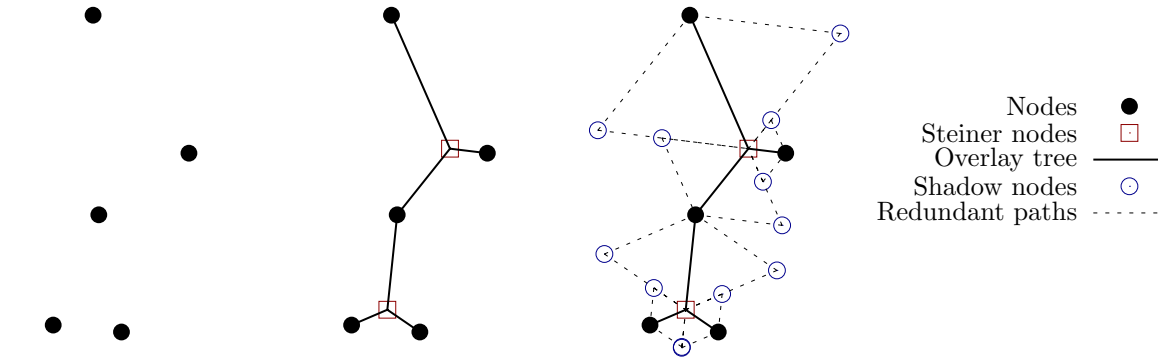
Solving the equation, we get $r = m \cdot d$ where

$$m = \frac{1 + \sqrt{1 + 4(10^{\frac{10\log 2 + \mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}}{5\alpha}} - 1)}}{2(10^{\frac{10\log 2 + \mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}}{5\alpha}} - 1)}$$

The result shows that the distance between nodes after the mesh branching point is proportional to the distance of the first hop. The result is similar to the result of branching nodes with three children presented in Section 5.2.2 if $\theta = \frac{\pi}{3}$. Figure 39 shows values of m at different $\mathrm{SIR}_{\mathrm{min}}^{\mathrm{dB}}$ and α.

6.1.2 Mesh merging nodes

Next, we consider a mesh merging node where the transmission from the redundant paths finally merge back to the overlay node. Since the length of the overlay path is l and the length of the redundant path is $2l$, the transmissions on the overlay path will have already passed through the overlay node by the time the two redundant paths converge to the merging node. The merging scenario is illustrated in Figure 38 (b).



(a) multicast group (b) a Steiner overlay tree (c) an overlay mesh

Figure 40: An example of OLE algorithm.

The analysis for the transmissions along the redundant paths is identical to the analysis of branching nodes with two children presented in Section 5.2.1. However, the last hop transmissions to the merging node cannot take place at the same time. There are two possible solutions to this problem

1. the transmissions to the merging node can be scheduled to avoid collision, and
2. the node may skip the transmission to the merging node if it overheard the overlay node previously transmitted the same multicast packet.

The second option is possible if the overlay node is not a leaf node in the multicast tree since the overlay node will not forward the packet if it is a leaf node in the multicast tree. An example of the overlay mesh extension algorithm is presented in Figure 40.

After the overlay mesh is formed, the algorithm proceeds to build a multicast mesh routing structure. The algorithm first finds a node that is nearest to each shadow node location and assigns the selected node as the shadow node. Finally, the algorithm connects overlay nodes and shadow nodes using the analyses for the mesh branching nodes and the mesh merging nodes presented earlier.

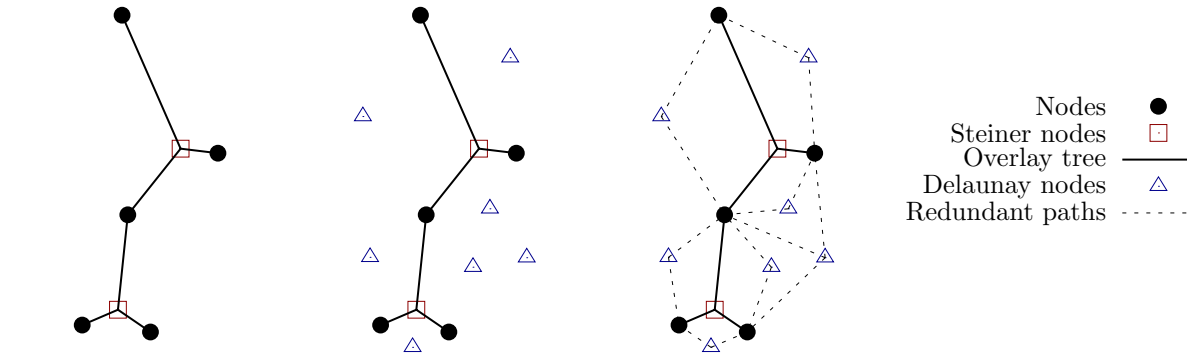
One advantage of the overlay link extension algorithm is that the analysis can be used to select actual nodes on the final overlay mesh. However, as can be seen in Figure 40, a noticeable problem with the overlay link extension algorithm is that the algorithm adds two redundant paths for all overlay links without considering other links. This simple approach can create overlapping redundant paths, which may create more interference to the point where the paths are no longer working reliably.

6.2 Delaunay mesh extension algorithm

As stated in Section 6.1, the overlay link extension algorithm does not consider the overall tree structure when extending a multicast tree to form a mesh, resulting in overlapping redundant paths. In this section, we propose our second algorithm to extend a multicast tree to form a mesh, called Delaunay mesh extension algorithm (DME). Our goal in designing the second algorithm is to take the *overall* tree structure into account when extending a multicast tree to form a mesh.

The idea of the Delaunay mesh extension algorithm is to use Delaunay triangulation on the set of overlay nodes in the overlay multicast tree to identify the positions of the mesh nodes (also called Delaunay nodes). Given a Delaunay triangulation, the algorithm identifies a center point of each triangle as a potential mesh node. The algorithm creates a redundant link between each potential mesh node and each of its corresponding overlay nodes if the overlay node is not a Steiner node. We do not create a redundant link between the potential mesh node and the Steiner node since the Steiner node is not a source or a receiver in the multicast group and does not need to be protected by a redundant path. In this case, the potential mesh node simply connects between two nodes in the multicast group. An example of the Delaunay mesh extension algorithm is presented in Figure 41.

One advantage of the Delaunay mesh extension is that the redundant paths will not overlap each other. One drawback of the Delaunay mesh extension is that the



(a) a Steiner overlay tree (b) Delaunay nodes (c) an overlay mesh

Figure 41: An example of DME algorithm.

analysis cannot be applied directly like the overlay link extension since the lengths of the redundant paths are not identical. To solve this problem, we use the same concept of the preferred scaling factor (f_p) as the fixed distance tree merging algorithm (FTM) presented in Section 5.3 when building the final multicast structure.

Even though the Delaunay mesh extension does not create overlapped redundant paths, it is still possible for two Delaunay nodes to be located close together, which will result in higher interference on the redundant paths. To solve this problem, we propose a variation of the Delaunay mesh extension algorithm in the next section.

6.2.1 Delaunay mesh extension with nodes merging

The goal of the Delaunay mesh extension with nodes merging algorithm (DME-merge) is to combine two Delaunay nodes that are located close together into one Delaunay node. The algorithm takes an extra argument called `merging_distance` that is used to determine if two Delaunay nodes should be merged into one Delaunay node.

The algorithm takes the mesh structure from the Delaunay mesh extension and repeatedly combines two Delaunay nodes if they are separated by a distance smaller than the given `merging_distance` and the straight line between the two nodes does not cross the original overlay tree. The algorithm merge two Delaunay nodes by

placing a new Delaunay nodes at the midpoint between the two Delaunay nodes. The new Delaunay node is now connected to all the overlay nodes that were originally connected to the two Delaunay nodes. The algorithm tries to merge two Delaunay nodes that satisfy the given condition until no two Delaunay nodes are separated by a distance shorter than the given `merging_distance`.

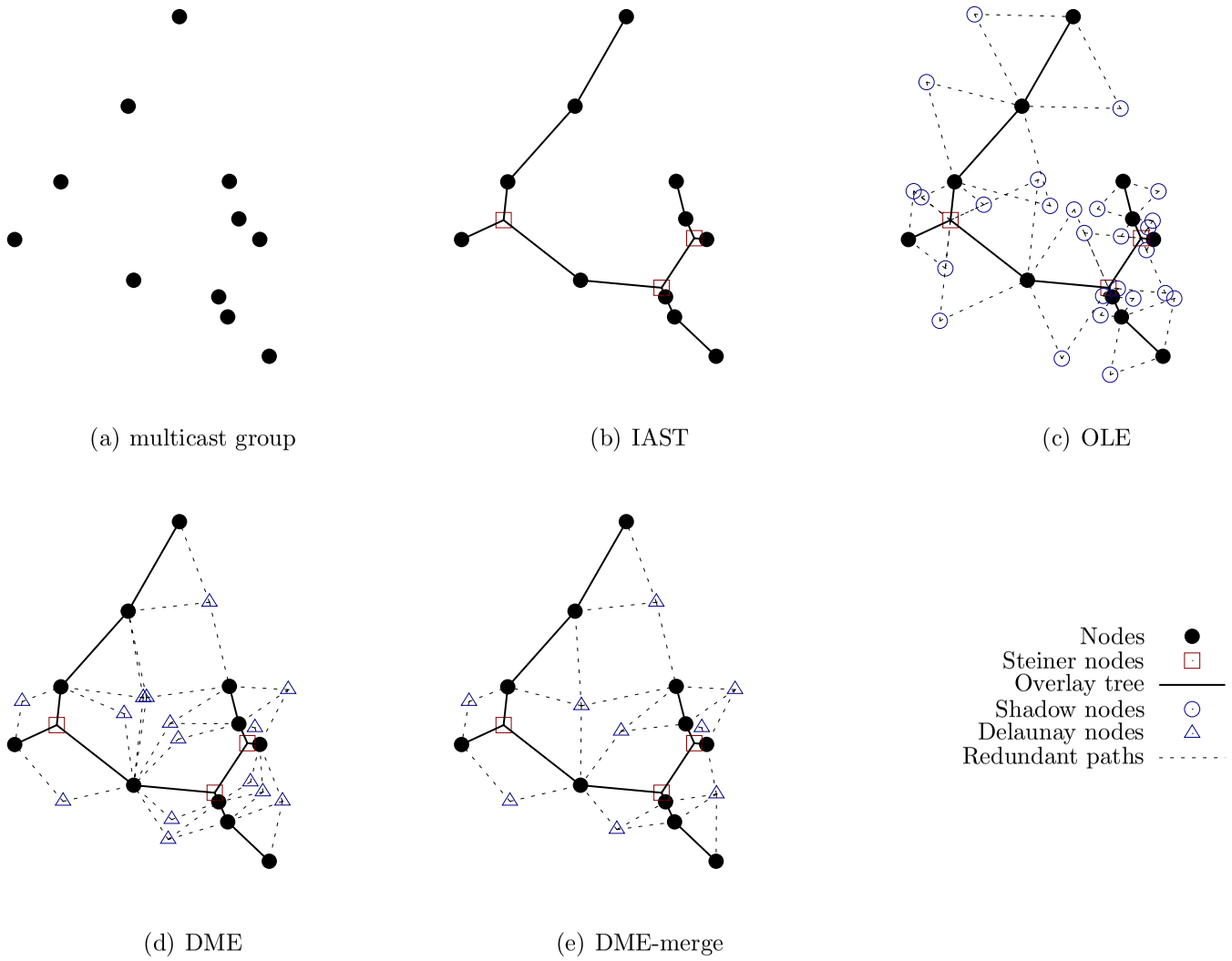
We present all interference-aware multicast routing structures in Figure 42. The multicast group (1 source and 10 destinations) is shown in Figure 42 (a), interference-aware Steiner multicast tree (IAST) is shown in Figure 42 (b), overlay link extension (OLE) is shown in Figure 42 (c), Delaunay mesh extension (DME) is shown in Figure 42 (d), and Delaunay mesh extension with node merging (DME-merge) is shown in Figure 42 (e).

6.3 Performance evaluation

We evaluate our algorithms in four sets of simulations. First, we start by evaluating the scaling factor to be used by the overlay link extension algorithm and Delaunay mesh extension algorithm. Second, we evaluate our proposed multicast mesh structures and other multicast routing structures in a network with bursty wireless links. We study the impact of the level of burstiness in the third set of simulations. Finally, we evaluate the performance of our algorithms in a network where nodes randomly drop multicast packets.

6.3.1 Simulation parameters and assumptions

We use ns-3.15 simulator to evaluate all algorithms. We use a physical model of 802.11g at the data rate of 6 Mbps in the simulation. All nodes use the transmission power of 40 mW and thermal noise is computed at 290K. All wireless links are modeled with ideal bursty link model with $b^+ = b^- = 0.2$ (see Section 3.2.2), unless stated otherwise. In all simulations, 2000 nodes were uniformly distributed in a deployment area of 1000 m by 1000 m. Common simulation parameters in Table 3 are used in



(a) multicast group (b) IAST (c) OLE

(d) DME (e) DME-merge

Figure 42: Examples of different interference-aware multicast routing structures.

Table 3: Simulation parameters used to evaluate tree extension structures.

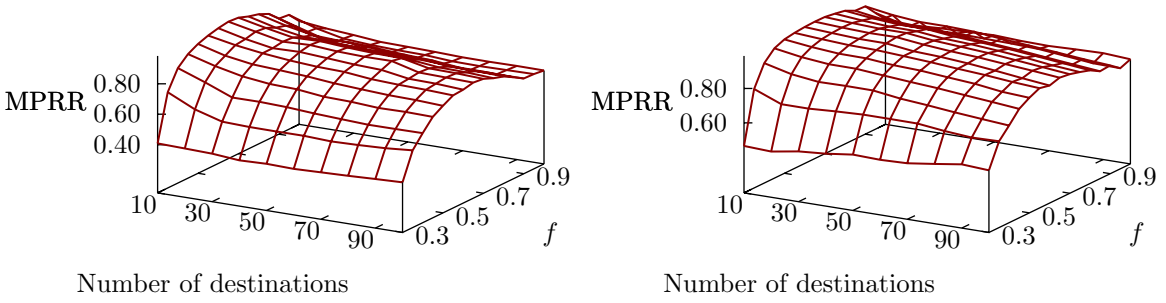
Parameter	**Value**
Deployment area	1000 m by 1000 m
Number of nodes	2000
Mobility model	constant position
Propagation loss model	log-distance
Path-loss exponent	3
Transmission power	40 mW
Thermal noise	290K
Burstiness model	ideal bursty
Device	IEEE 802.11g
Default transmission mode	6 Mbps

all simulations, unless stated otherwise. All results reported are averaged from 100 simulations.

6.3.2 Tuning scaling factor and merging distance

We first evaluate the scaling factor since it is a significant parameter affecting both OLE and DME. In this simulation, one source node is randomly selected as a multicast source. The source node sends multicast packet at the rate of 10 packets per second. We vary the scaling factor from 0.3 to 1.0 and collect the multicast packet reception ratio of both OLE and DME. We did not use the scaling factor below 0.3 since the network became disconnected in some simulations. The simulation results are reported in Figure 43.

As seen from Figure 43, the choice of scaling factor affects the performance of OLE and DME algorithms. If the scaling factor is too small, the extra nodes included create more interference that can outweigh the gain of spatial reuse. If the scaling factor is too large, the links are prone to interference from other concurrent transmissions. The optimal scaling factor is also dependent on the number of multicast destinations in the network as the number of nodes in the multicast trees changes. However,



(a) Overlay link extension

(b) Delaunay mesh extension

Figure 43: MPRR of OLE and DME with different scaling factors.

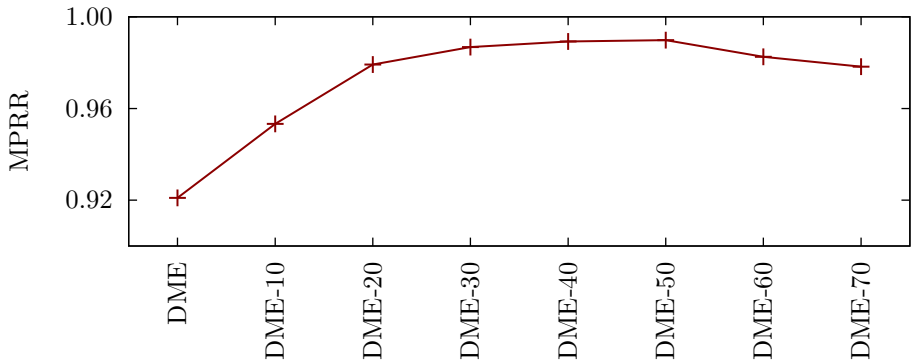


Figure 44: MPRR of DME-merge algorithm with different merging distance.

performance is quite stable across a fairly wide range of scaling factors, e.g. 0.5 to 0.7. Based on this analysis, we have set the scaling factor to 0.7 in the remaining simulations.

Next, we evaluate the merging distance parameter of DME-merge algorithm. In this simulation, we use the scaling factor of 0.7 and vary the merging distance of DME-merge from 10 m to 70 m. Multicast packet reception ratios of DME-merge are reported in Figure 44.

Figure 44 confirms that the choice of merging distance affects the performance of DME-merge algorithm. When the merging distance is small, only a few Delaunay nodes are merged together, the mesh structure of DME-merge is still similar to the mesh structure of DME. However, if the merging distance is large, DME-merge will

aggressively merge Delaunay nodes. This aggressive merging can result in reduced performance. When DME-merge merges two Delaunay nodes, the resulting Delaunay node is responsible for all links of the two original Delaunay nodes. Thus, the more Delaunay nodes merged, the more links the new Delaunay node must take care of. In an extreme case, this can create a Delaunay node that is connected to all other overlay nodes, while having only one incoming path to the Delaunay node.

Again, performance is quite stable across a fairly wide range of merging distance, e.g. 30 to 50. Based on this results, we have set the merging distance to 50 in the remaining simulations.

6.3.3 Performance of multicast routing structures with bursty links

In this simulation, we evaluate the performance of different multicast routing structures when the wireless links exhibit bursty behavior. Wireless communications in this simulation are modeled with ideal bursty link, using $b^+ = b^- = 0.2$. The number of multicast destinations was varied from 10 to 100. A single multicast source was randomly selected among the remaining nodes. The source node sends multicast packets at the rate of 10 packets per second for 600 seconds. We have implemented another mesh multicast routing structure called MDM for comparison [101]. We measured the multicast packet reception ratio (MPRR) and report the results in Figure 45.

As seen from Figure 45, the performance of different multicast routing structures are different. The performance of the shortest path tree is the lowest among the multicast routing structures. Since the shortest path tree does not consider interference when building a tree, it is more prone to interference and bursty links than other structures. The tree structure also makes it vulnerable to even a single transmission failure as the tree will be disconnected.

Our interference-aware Steiner tree provides improvement over the shortest path tree since IAST takes interference into account when building a tree. As a result,

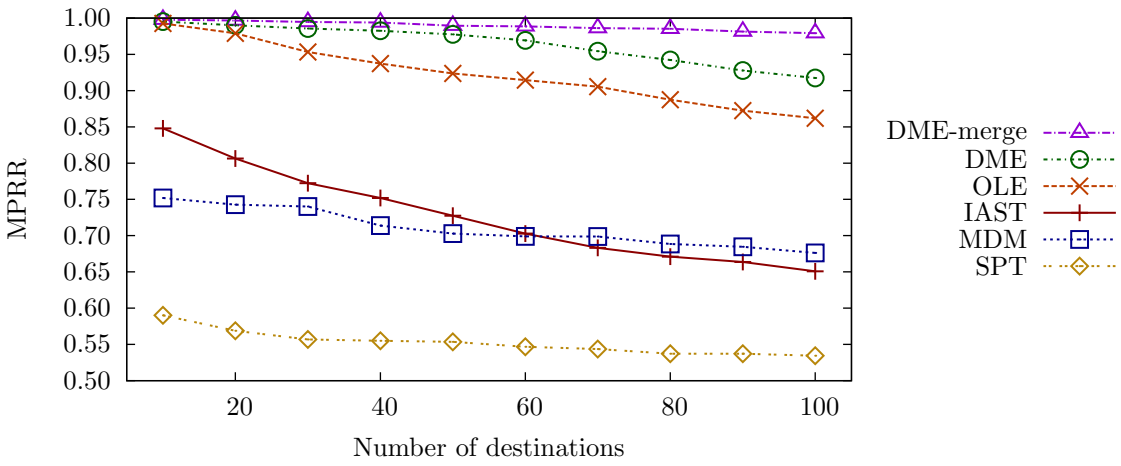


Figure 45: MPRR of different mesh multicast structures.

IAST is less prone to interference and bursty links than the shortest path tree. Still, a single transmission failure will disconnect the multicast tree of IAST. MDM also provides improvement over the shortest path tree by including multiple paths to deliver multicast packets to the destinations. This robustness increases the performance of the multicast structure since a single transmission failure will not result in receivers missing the multicast packets. However, since MDM does not take interference into account when building a mesh, it is still prone to transmission failure along the mesh.

OLE and DME algorithms have higher multicast packet reception ratios than other multicast routing structures, including our previously proposed interference-aware Steiner tree algorithm. The additional paths included by extending the Steiner overlay tree allow the multicast packets to be delivered to the destinations even if some multicast packets were dropped on other paths to the receivers. Moreover, the paths built by both OLE and DME algorithms are interference-aware, which means that the nodes on the paths are less prone to transmission failure than other multicast mesh routing structures.

Among our three mesh algorithms, Delaunay mesh extension with nodes merging provides the highest multicast packet reception ratio, especially at the higher number of multicast destinations. When the number of multicast destinations is high, the

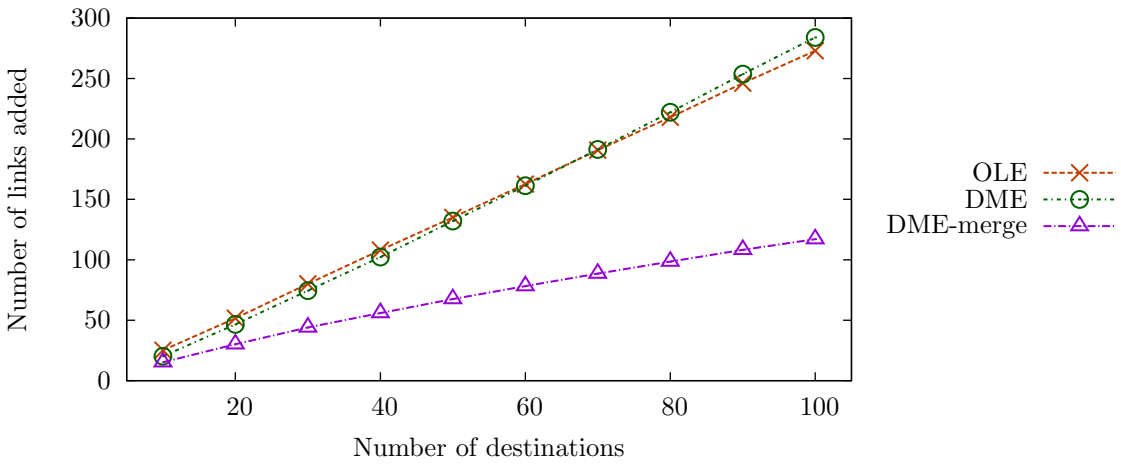


Figure 46: Number of links added to the multicast tree from different algorithms.

number of links that are introduced by the overlay link extension algorithm and Delaunay mesh extension algorithm increase quickly. The number of links added to the multicast tree after different tree extension algorithms were applied to the multicast tree are reported in Figure 46.

As seen from Figure 46, the number of links added to the multicast tree increases as the number of destinations increases. The added links can create more contention in the network and introduce more interference among the paths. This effect results in lower multicast packet reception ratios of the overlay link extension algorithm and Delaunay mesh extension algorithm. For Delaunay extension algorithm with nodes merging, the algorithm tries to merge Delaunay nodes that are located close together, which reduces the number of new links in the final multicast structure. As a result, Delaunay extension algorithm with nodes merging does not suffer from this problem as much as the other two tree extension algorithms.

6.3.4 Link burstiness and its impact

In this set of simulations, we study the impact of the burstiness of the wireless links on the performance of multicast routing structures. Again, we varied the number of multicast destinations from 10 to 100 with one node selected randomly as a multicast

source. We varied the wireless link burstiness by varying the parameter b^+ and b^- of the ideal bursty link model from 0.1 to 0.5. Multicast packet reception ratios of different multicast routing structures are reported in Figure 47.

As seen from Figure 47, MDM is more sensitive to the level of link burstiness than our proposed multicast routing structures. Multicast packet reception ratio of MDM decreases as the level of link burstiness increases. For our proposed multicast routing structures, the performance is relatively stable when burstiness level changes.

6.3.5 Performance of multicast routing structures with faulty nodes

In this set of simulations, we look at another cause for disconnected graph – a presence of faulty nodes in the network. For simplicity, we assume that wireless links are not bursty in this set of simulations. To simulate a faulty node, each node that is participating in the multicast routing structure randomly decides to drop a multicast packet instead of forwarding the packet to the next nodes. The probability of dropping a multicast packet was varied from 0.02 to 0.10. The decision to drop the packet is made independently for each multicast packet. In this set of simulations, the number of multicast destinations was kept constant at 50. Multicast packet reception ratios of different multicast routing structures are reported in Figure 48.

As seen from Figure 48, the performance of most multicast routing structures drops as the probability of faulty node increases. The performance drop is substantial for IAST since it relies on a tree as a multicast routing structure. A single faulty node along the tree will disconnects the subtree below the faulty node.

For MDM, the multicast packet reception ratio drops slightly from about 0.80 to about 0.70. MDM relies on mesh structure, which makes it more robust when a few nodes are faulty. However, the number of redundant paths of MDM is not large enough to handle a large number of faulty nodes, which results in slight drop in multicast packet reception ratio when the probability of faulty node is high.

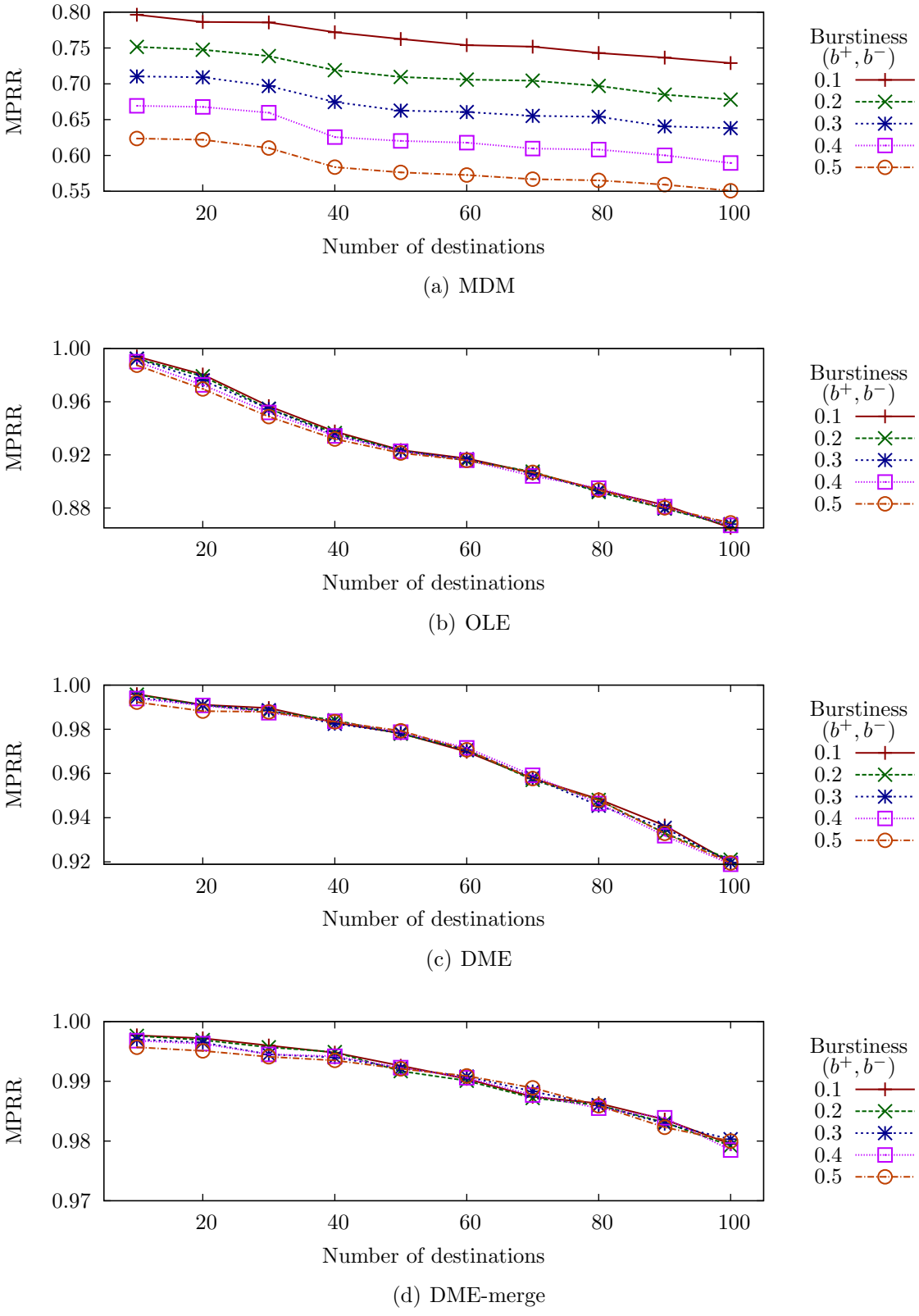


Figure 47: MPRR of multicast routing structures with different burstiness.

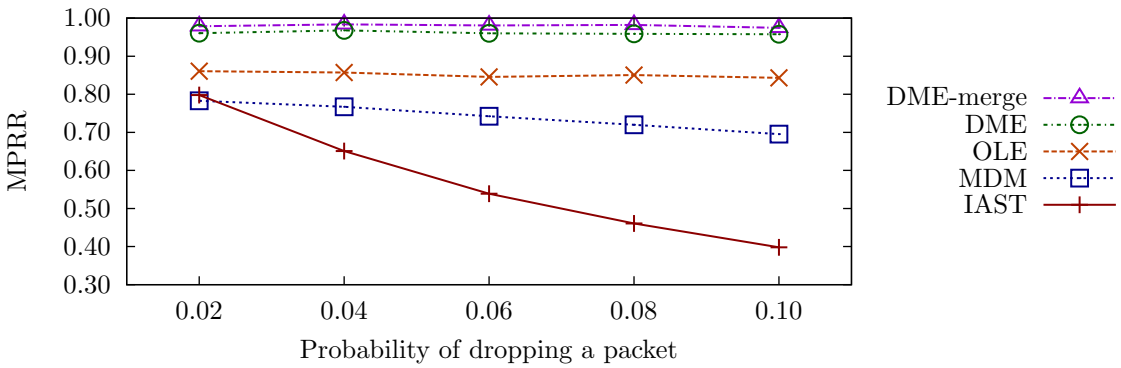


Figure 48: MPRR of different algorithms at varying faulty node probability.

Our proposed mesh multicast routing structures can withstand a larger number of faulty nodes than MDM and IAST as can be seen by the almost constant multicast packet reception ratios even at a high faulty probability. The extra paths included by OLE, DME, and DME-merge make them more robust to faulty nodes than other multicast routing structures.

6.4 Discussion

In this chapter, we have proposed two algorithms to extend the interference-aware Steiner multicast tree to form an interference-aware multicast mesh. The main idea of both algorithms is to include a set of shadow nodes that are connected back to the overlay tree to form an overlay mesh. Once an overlay mesh is formed, the algorithm builds the actual multicast mesh structure using the overlay mesh as a guideline. We have evaluated the performance of our proposed algorithms in four different settings and showed that our proposed algorithms provide higher multicast packet reception ratios than other multicast mesh routing structures that do not consider interference when building the mesh. The performance of our proposed algorithms is also relatively stable across different levels of wireless links burstiness or node failure rate.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this dissertation, we have studied the problem of multicasting in wireless multihop networks. In this final chapter, we summarize the contributions of this work and discuss possible future research direction.

7.1 Contributions

In Chapter 3, we proposed a model to simulate wireless link burstiness. The model works by keeping track of the history of a wireless link and makes adjustment to the probability of successfully receiving the next transmission based on the history of the link. We implemented the proposed model in the ns-3 network simulator and showed that our proposed bursty link model is able to accurately simulate wireless link burstiness observed in real wireless links. We have shown that link burstiness affects the performance of DSR routing protocol.

In Chapter 4, we studied the problem of multicasting at the MAC layer. We proposed an extension to the IEEE 802.11 with the goals to provide reliability for multicast transactions and to integrate neighbor discovery mechanism to the MAC layer. We have shown that our extension is able to quickly and accurately detect neighborhood change at the MAC layer and also provides higher reliability for multicast transaction than other MAC layer multicast protocols.

In Chapter 5, we moved up the stack and studied the problem of multicasting at the network-wide level. We classified nodes in multicast trees into different classes and showed, by analyzing an accurate physical interference model, that different classes require different routing strategies. Based on the analyses, we proposed two joint routing and scheduling algorithms for multicasting in wireless multihop networks.

We evaluated the performance of different algorithms through simulation and showed that the proposed algorithms outperform other algorithms both in terms of schedule lengths and goodput. We ended the chapter by showing the impact of wireless link burstiness on the proposed multicast tree structures.

Based on the results of the study at the end of Chapter 5, we considered the problem of building an interference-aware mesh multicast routing structure in Chapter 6. We proposed two algorithms to extend an interference-aware multicast tree to form an interference-aware multicast mesh based on the interference analyses. We showed that the proposed interference-aware mesh structures provide higher reliability than our multicast tree structure and other mesh-based multicast structures, even with the presence of bursty wireless links or faulty nodes in the network.

7.2 Future work

In this study, we have proposed a number of models and ideas to solve the problem of multicasting in wireless multihop networks. In addition to the contributions, our work also opens up a number of topics for future research.

7.2.1 Simulating frame-level bursty links in wireless networks

Even though we have shown that our stochastic bursty-link model is able to simulate variety of bursty characteristics observed in real wireless links, the research in this area is still far from complete. Further refinements to our model are possible. In our current model, all bursts with the same size are treated equally if the bursts occurred within the cache TTL. Further refinement to assign different weights depending on the actual timing that the bursts occurred within the cache TTL is a very interesting area to explore. The goal is to give more recent bursts higher impact on the probability of successfully receiving the next transmission than bursts that occurred in the past.

The methodology to obtain a trace file can also be further improved. For example, changing from broadcasting packets to unicasting packets is likely to yield different

link bursty characteristics due to the interaction between MAC layer retransmission and back-off mechanisms. A bursty link model designed specifically to handle link burstiness in unicast transaction can be created.

Our bursty link model can be considered a short-term bursty link model. Integrating our bursty link model with other long-term models such as auto-regression model for wireless channel variation is a possibility.

Lastly, improving the memory complexity of our bursty link model could increase the efficiency and reduce running time of our bursty link model, especially in a large network simulation.

7.2.2 MAC layer reliable multicast and link discovery

We have proposed an extension to IEEE 802.11 to provide reliability to multicast transaction and to integrate neighbor discovery mechanism at the MAC layer. As we showed in Chapter 4, the overhead of the positive acknowledgement mechanism grows quickly as the number of neighbors increases. One possible future research topic would be to incorporate block ACK mechanism to our proposed extension. By using block ACK mechanism, the overhead of the positive acknowledgement can be reduced, which will result in a more efficient use of wireless spectrum. Another possible research topic would be the application of the concept of Join ACK to unicast transactions.

7.2.3 Interference-aware multicast

We have carried out analyses of a branching node with two children and a branching node with three children, using the log-distance propagation loss model. One possible area worth exploring would be to perform the same analysis using other propagation loss models. Moreover, the analysis we carried out did not consider the fading effect or background noise. For example, a log-normal fading that models long term fading could be added into the inter-node interference model.

Finally, our analyses of interference-aware multicast tree and mesh structures gave us valuable insight about the importance of considering interference when constructing a multicast routing structure in an ideal setting. Designing a practical multicast protocol that utilizes the analyses that we carried out to construct an interference-aware multicast tree or mesh constitutes a possible future research direction.

APPENDICES

APPENDIX A

TRACE FILES INFORMATION

In this appendix, we list the information about the 100 trace files we used for the simulations in Chapter 3. The duration of all trace files are one hour.

Table 4: Trace files information.

File ID	Wi-Fi mode	Channel	Wi-Fi device	Start time	β
1	6 Mbps	1	Atheros	2013-08-31 16:43	0.8313
2	6 Mbps	1	Atheros	2013-08-31 16:44	0.8102
3	36 Mbps	1	Atheros	2013-09-02 09:13	0.7896
4	36 Mbps	1	Atheros	2013-09-02 10:15	0.7616
5	36 Mbps	1	Atheros	2013-09-02 07:07	0.7552
6	36 Mbps	1	RaLink	2013-09-01 16:43	0.7514
7	6 Mbps	1	RaLink	2013-08-31 16:43	0.7472
8	36 Mbps	1	Atheros	2013-09-02 07:06	0.7434
9	36 Mbps	1	RaLink	2013-09-01 15:36	0.7352
10	6 Mbps	6	Atheros	2013-10-23 12:00	0.7247
11	36 Mbps	1	Atheros	2013-09-02 09:13	0.7233
12	36 Mbps	1	Atheros	2013-09-02 08:10	0.7191
13	36 Mbps	1	RaLink	2013-09-01 17:48	0.6767
14	36 Mbps	1	Atheros	2013-09-01 15:36	0.6667
15	36 Mbps	1	Atheros	2013-09-02 08:10	0.6646
16	36 Mbps	1	RaLink	2013-09-02 10:14	0.6621

(Continued on next page)

Table 4 (continued)

File ID	Wi-Fi mode	Channel	Wi-Fi device	Start time	β
17	36 Mbps	1	Atheros	2013-09-01 17:48	0.6582
18	6 Mbps	6	RaLink	2013-08-29 08:38	0.6558
19	6 Mbps	6	RaLink	2013-08-29 08:52	0.6553
20	6 Mbps	6	RaLink	2013-08-29 20:42	0.6544
21	36 Mbps	1	Atheros	2013-09-01 16:43	0.6527
22	36 Mbps	1	RaLink	2013-09-02 08:10	0.6504
23	6 Mbps	1	Atheros	2013-08-31 17:50	0.6464
24	6 Mbps	6	RaLink	2013-10-23 12:00	0.6456
25	36 Mbps	1	RaLink	2013-09-02 09:13	0.6407
26	6 Mbps	6	RaLink	2013-10-23 12:00	0.6397
27	6 Mbps	6	RaLink	2013-08-28 17:12	0.6367
28	6 Mbps	6	RaLink	2013-08-28 17:12	0.6353
29	36 Mbps	1	RaLink	2013-09-02 07:06	0.6342
30	6 Mbps	1	RaLink	2013-08-31 17:50	0.6331
31	36 Mbps	1	RaLink	2013-09-01 15:36	0.6329
32	6 Mbps	6	RaLink	2013-08-28 16:09	0.6265
33	6 Mbps	1	Atheros	2013-08-31 17:50	0.6259
34	6 Mbps	6	RaLink	2013-10-23 12:00	0.6244
35	36 Mbps	1	Atheros	2013-09-02 10:14	0.6202
36	6 Mbps	6	RaLink	2013-08-29 12:52	0.6195
37	54 Mbps	1	RaLink	2013-08-31 18:58	0.6186
38	6 Mbps	6	RaLink	2013-08-28 14:37	0.6112
39	6 Mbps	6	RaLink	2013-08-28 12:23	0.6102

(Continued on next page)

Table 4 (continued)

File ID	Wi-Fi mode	Channel	Wi-Fi device	Start time	β
40	36 Mbps	1	RaLink	2013-09-02 07:07	0.5923
41	6 Mbps	6	RaLink	2013-08-28 16:09	0.5909
42	6 Mbps	6	RaLink	2013-08-29 08:38	0.5759
43	36 Mbps	1	RaLink	2013-09-02 15:29	0.5642
44	36 Mbps	1	RaLink	2013-09-02 09:13	0.5615
45	36 Mbps	1	RaLink	2013-09-02 10:15	0.5530
46	36 Mbps	1	RaLink	2013-09-02 08:10	0.5472
47	36 Mbps	1	Atheros	2013-09-02 15:29	0.5466
48	6 Mbps	6	RaLink	2013-10-23 12:00	0.5464
49	36 Mbps	1	Atheros	2013-09-02 13:22	0.5383
50	36 Mbps	1	Atheros	2013-09-02 15:29	0.5303
51	36 Mbps	1	RaLink	2013-09-01 17:48	0.5293
52	36 Mbps	1	RaLink	2013-09-02 13:21	0.5282
53	6 Mbps	6	RaLink	2013-08-28 14:37	0.5266
54	36 Mbps	1	Atheros	2013-09-02 13:21	0.5218
55	54 Mbps	1	Atheros	2013-08-31 18:58	0.5121
56	36 Mbps	1	RaLink	2013-09-01 16:43	0.5090
57	36 Mbps	1	RaLink	2013-09-02 14:24	0.4951
58	54 Mbps	1	Atheros	2013-09-01 13:20	0.4903
59	54 Mbps	1	RaLink	2013-09-01 14:29	0.4846
60	36 Mbps	1	Atheros	2013-09-02 14:24	0.4789
61	36 Mbps	1	Atheros	2013-09-02 14:24	0.4780
62	36 Mbps	1	RaLink	2013-09-02 15:29	0.4751

(Continued on next page)

Table 4 (continued)

File ID	Wi-Fi mode	Channel	Wi-Fi device	Start time	β
63	6 Mbps	1	RaLink	2013-08-31 17:50	0.4700
64	6 Mbps	1	RaLink	2013-08-31 16:44	0.4602
65	36 Mbps	1	Atheros	2013-09-01 17:48	0.4559
66	36 Mbps	1	RaLink	2013-09-02 14:24	0.4537
67	36 Mbps	1	Atheros	2013-09-01 15:36	0.4528
68	36 Mbps	1	RaLink	2013-09-02 13:22	0.4453
69	54 Mbps	1	RaLink	2013-09-01 13:20	0.4428
70	54 Mbps	1	Atheros	2013-09-01 14:29	0.4425
71	6 Mbps	6	RaLink	2013-08-29 12:52	0.4296
72	54 Mbps	1	RaLink	2013-08-31 18:58	0.4136
73	6 Mbps	1	Atheros	2013-09-08 16:38	0.4134
74	6 Mbps	1	Atheros	2013-09-08 13:14	0.4024
75	6 Mbps	1	Atheros	2013-09-08 19:43	0.4023
76	6 Mbps	1	Atheros	2013-09-08 07:31	0.3923
77	6 Mbps	1	Atheros	2013-09-08 10:06	0.3776
78	6 Mbps	1	Atheros	2013-09-08 17:59	0.3720
79	6 Mbps	1	Atheros	2013-09-08 08:52	0.3685
80	6 Mbps	1	Atheros	2013-09-08 14:18	0.3594
81	6 Mbps	1	Atheros	2013-09-08 10:59	0.3555
82	6 Mbps	1	Atheros	2013-09-08 14:18	0.3521
83	6 Mbps	1	Atheros	2013-09-08 15:29	0.3481
84	6 Mbps	1	Atheros	2013-09-08 13:14	0.3377
85	6 Mbps	1	Atheros	2013-09-08 10:59	0.3355

(Continued on next page)

Table 4 (continued)

File ID	Wi-Fi mode	Channel	Wi-Fi device	Start time	β
86	6 Mbps	1	Atheros	2013-09-08 20:58	0.3203
87	6 Mbps	1	Atheros	2013-09-08 08:52	0.3166
88	6 Mbps	1	Atheros	2013-09-08 10:06	0.3159
89	6 Mbps	1	Atheros	2013-09-08 17:59	0.3147
90	6 Mbps	1	Atheros	2013-09-08 16:38	0.3145
91	6 Mbps	1	Atheros	2013-09-08 20:58	0.3072
92	6 Mbps	1	Atheros	2013-09-08 19:43	0.2999
93	6 Mbps	1	Atheros	2013-09-08 07:31	0.2992
94	6 Mbps	1	Atheros	2013-09-08 12:07	0.2958
95	6 Mbps	1	Atheros	2013-09-08 12:07	0.2944
96	6 Mbps	1	Atheros	2013-09-08 15:29	0.2900
97	54 Mbps	1	Atheros	2013-09-09 08:48	0.2812
98	54 Mbps	1	Atheros	2013-09-09 08:48	0.2779
99	54 Mbps	1	Atheros	2013-09-09 10:15	0.2437
100	54 Mbps	1	Atheros	2013-09-09 10:15	0.2402

APPENDIX B

BURSTY LINK SIMULATION RESULTS

We report the full simulation results from the 100 trace files in this Appendix. All results from the 100 trace files are reported in Table 5 using the burstiness metric (β) proposed by Srinivasan et al. [83]. The column named Trace file shows the results from analyzing the trace file. The column named Discrete BPA shows the results from simulating the link using a trace file-based bursty model (Section 3.2.1). The column named Scaled-erf shows the results from simulating the link using the scaled-erf function (Section 3.2.2). The column named BEAR shows the results from modeling the link with BEAR model [32]. The last column, ns-3, shows the results from modeling the link with the default ns-3 model. All results from simulated links were averaged from 1000 simulations.

Table 5: Burstiness metrics of the 100 trace files.

File ID	Trace file	Discrete BPA	Scaled-erf	BEAR	ns-3
1	0.8313	0.8103	0.7914	0.2602	-0.0011
2	0.8102	0.7464	0.7265	0.2611	-0.0004
3	0.7896	0.7586	0.7357	0.2986	0.0001
4	0.7616	0.6949	0.6653	0.2597	0.0009
5	0.7552	0.7747	0.7878	0.2972	0.0001
6	0.7514	0.7676	0.7813	0.2458	0.0007
7	0.7472	0.6507	0.6280	0.2283	-0.0011
8	0.7434	0.8224	0.8387	0.2891	0.0004

(Continued on next page)

Table 5 (continued)

File ID	Trace file	Discrete BPA	Scaled-erf	BEAR	ns-3
9	0.7352	0.7346	0.7469	0.2569	-0.0013
10	0.7247	0.7496	0.7643	0.2518	-0.0001
11	0.7233	0.7646	0.7818	0.2467	0.0010
12	0.7191	0.7614	0.7760	0.2868	-0.0013
13	0.6767	0.6651	0.6841	0.3366	0.0004
14	0.6667	0.6554	0.6712	0.3578	0.0003
15	0.6646	0.6476	0.6771	0.3657	0.0003
16	0.6621	0.6498	0.6660	0.3255	0.0006
17	0.6582	0.6401	0.6626	0.3369	0.0006
18	0.6558	0.6355	0.6652	0.3153	0.0010
19	0.6553	0.6377	0.6638	0.3379	0.0007
20	0.6544	0.6438	0.6595	0.2757	0.0007
21	0.6527	0.6385	0.6584	0.2852	0.0003
22	0.6504	0.6370	0.6583	0.3001	0.0001
23	0.6464	0.6302	0.6548	0.3615	0.0007
24	0.6456	0.6311	0.6551	0.3040	0.0002
25	0.6407	0.6249	0.6493	0.3375	0.0020
26	0.6397	0.6222	0.6477	0.2979	0.0009
27	0.6367	0.6228	0.6433	0.3104	0.0001
28	0.6353	0.6176	0.6440	0.3698	0.0006
29	0.6342	0.6154	0.6464	0.3632	0.0003
30	0.6331	0.6397	0.6524	0.2543	-0.0003
31	0.6329	0.6213	0.6388	0.3483	0.0007

(Continued on next page)

Table 5 (continued)

File ID	Trace file	Discrete BPA	Scaled-erf	BEAR	ns-3
32	0.6265	0.6371	0.6538	0.2860	-0.0010
33	0.6259	0.6068	0.6380	0.2769	0.0005
34	0.6244	0.6148	0.6335	0.2886	0.0002
35	0.6202	0.6152	0.6328	0.3098	0.0009
36	0.6195	0.6030	0.6339	0.3139	0.0007
37	0.6186	0.6145	0.6260	0.2972	0.0008
38	0.6112	0.6059	0.6211	0.3095	0.0009
39	0.6102	0.6485	0.6641	0.2778	-0.0001
40	0.5923	0.6578	0.7028	0.2783	-0.0006
41	0.5909	0.5906	0.6022	0.2658	0.0009
42	0.5759	0.5734	0.6000	0.2871	-0.0017
43	0.5642	0.5936	0.6148	0.2650	-0.0009
44	0.5615	0.5900	0.6051	0.2666	-0.0001
45	0.5530	0.5532	0.5579	0.2488	-0.0012
46	0.5472	0.5489	0.5722	0.2755	-0.0004
47	0.5466	0.6168	0.6480	0.2821	-0.0017
48	0.5464	0.5710	0.5891	0.2740	-0.0006
49	0.5383	0.5509	0.5670	0.2713	0.0000
50	0.5303	0.5308	0.5449	0.2685	0.0006
51	0.5293	0.5299	0.5436	0.2838	-0.0009
52	0.5282	0.5688	0.5906	0.2464	0.0013
53	0.5266	0.5034	0.5065	0.3075	-0.0016
54	0.5218	0.4635	0.4513	0.2838	-0.0017

(Continued on next page)

Table 5 (continued)

File ID	Trace file	Discrete BPA	Scaled-erf	BEAR	ns-3
55	0.5121	0.4450	0.4340	0.2739	-0.0013
56	0.5090	0.5644	0.5923	0.2668	0.0035
57	0.4951	0.4949	0.4790	0.2581	-0.0011
58	0.4903	0.5314	0.5512	0.2733	-0.0003
59	0.4846	0.4691	0.4747	0.2775	-0.0002
60	0.4789	0.4069	0.3982	0.2816	-0.0001
61	0.4780	0.4754	0.4575	0.2716	-0.0018
62	0.4751	0.4794	0.4963	0.2792	-0.0009
63	0.4700	0.5158	0.5385	0.2558	-0.0015
64	0.4602	0.4623	0.4841	0.2912	-0.0017
65	0.4559	0.4995	0.5350	0.2817	-0.0006
66	0.4537	0.4536	0.4719	0.2982	-0.0017
67	0.4528	0.4646	0.4880	0.2695	-0.0010
68	0.4453	0.5119	0.5310	0.2791	-0.0012
69	0.4428	0.5104	0.5381	0.2722	-0.0020
70	0.4425	0.4425	0.4017	0.2650	-0.0014
71	0.4296	0.4765	0.5358	0.2464	-0.0018
72	0.4136	0.4130	0.3927	0.2537	-0.0010
73	0.4134	0.3859	0.3585	0.2849	-0.0007
74	0.4024	0.4126	0.4327	0.2588	0.0009
75	0.4023	0.4019	0.4002	0.2565	0.0007
76	0.3923	0.4037	0.4273	0.2657	-0.0012
77	0.3776	0.3775	0.3693	0.2624	-0.0007

(Continued on next page)

Table 5 (continued)

File ID	Trace file	Discrete BPA	Scaled-erf	BEAR	ns-3
78	0.3720	0.3719	0.4011	0.2405	-0.0013
79	0.3685	0.3687	0.3522	0.2645	0.0001
80	0.3594	0.3505	0.2515	0.2615	-0.0017
81	0.3555	0.3555	0.3581	0.2908	-0.0005
82	0.3521	0.3522	0.3442	0.2772	-0.0018
83	0.3481	0.3482	0.3626	0.2549	0.0025
84	0.3377	0.3376	0.3419	0.2710	-0.0008
85	0.3355	0.3370	0.3829	0.2727	0.0035
86	0.3203	0.3247	0.3402	0.2774	0.0025
87	0.3166	0.3158	0.3369	0.2701	0.0011
88	0.3159	0.3163	0.3097	0.2722	0.0037
89	0.3147	0.3148	0.3047	0.2870	-0.0002
90	0.3145	0.3155	0.3493	0.2685	-0.0013
91	0.3072	0.3076	0.3281	0.2633	-0.0001
92	0.2999	0.2997	0.3069	0.2581	0.0011
93	0.2992	0.3027	0.3583	0.2644	-0.0011
94	0.2958	0.2958	0.3099	0.2843	-0.0013
95	0.2944	0.2664	0.2545	0.2542	0.0007
96	0.2900	0.2900	0.2981	0.2840	0.0005
97	0.2812	0.2812	0.2638	0.2575	-0.0001
98	0.2779	0.2799	0.3106	0.2631	-0.0006
99	0.2437	0.2421	0.2956	0.2811	0.0008
100	0.2402	0.2400	0.2403	0.2672	-0.0001

APPENDIX C

PUBLICATIONS

The following publications resulted from this research.

- Daniel Lertpratchya, George F. Riley, and Douglas M. Blough, "Simulating frame- level bursty links in wireless networks," in Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques, March 2014 [55].

- Daniel Lertpratchya, George F. Riley, and Douglas M. Blough, "Fast and accurate link discovery integrated with reliable multicast in 802.11," Technical Report, Center for Experimental Research in Computer Systems, Georgia Institute of Technology, 2013 [54].

- Daniel Lertpratchya, Douglas M. Blough, and George F. Riley, "Interference-aware multicast for wireless multihop networks," in IEEE Wireless Communications and Networking Conference (WCNC), April 2014 [53].

- Daniel Lertpratchya, Douglas M. Blough, and George F. Riley, "Interference-aware mesh multicast for wireless multihop networks," to appear

REFERENCES

[1] "AirCrack-ng."
http://www.aircrack-ng.org.
Accessed: 2014-03-07.

[2] "Geosteiner - software for computing steiner trees."
http://www.diku.dk/hjemmesider/ansatte/martinz/geosteiner/.
Accessed: 2014-03-07.

[3] "MediaTek."
http://www.mediatek.com.
Accessed: 2014-03-07.

[4] "The ns-2 network simulator."
http://nsnam.isi.edu/nsnam.
Accessed: 2014-03-07.

[5] "The ns-3 network simulator."
http://www.nsnam.org.
Accessed: 2014-03-07.

[6] "Qualcomm Atheros."
http://www.atheros.com.
Accessed: 2014-03-07.

[7] "Wireless tools for linux."
http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html.
Accessed: 2014-03-12.

[8] "Wireless LAN medium access control (MAC) and physical layer (PHY) specification, amendment 4: Further higher data rate extension in the 2.4GHz band," 2003.

[9] "Specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 2: MAC enhancements for robust audio video streaming," 2012.

[10] Aguayo, D., Bicket, J., Biswas, S., Judd, G., and Morris, R., "Link-level measurements from an 802.11b mesh network," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '04, (New York, NY, USA), pp. 121–132, ACM, 2004.

[11] Basalamah, A., Sugimoto, H., and Sato, T., "Rate adaptive reliable multicast mac protocol for WLANs," in *Proceedings of the IEEE 63rd Vehicular Technology Conference*, vol. 3 of *VTC 2006-Spring*, pp. 1216–1220, May 2006.

[12] Blough, D., Resta, G., and Santi, P., "Approximation algorithms for wireless link scheduling with SINR-based interference," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1701–1712, Dec 2010.

[13] Brar, G., Blough, D. M., and Santi, P., "Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks," in *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, (New York, NY, USA), pp. 2–13, ACM, 2006.

[14] Campolo, C., Molinaro, A., Casetti, C., and Chiasserini, C., "An 802.11-based MAC protocol for reliable multicast in multihop networks," in *Proceedings of the IEEE 69th Vehicular Technology Conference*, VTC Spring 2009, pp. 1–5, April 2009.

[15] Cerpa, A., Wong, J. L., Potkonjak, M., and Estrin, D., "Temporal properties of low power wireless links: Modeling and implications on multi-hop routing," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '05, (New York, NY, USA), pp. 414–425, ACM, 2005.

[16] Chakeres, I. and Belding-Royer, E., "The utility of HELLO messages for determining link connectivity," in *The 5th International Symposium on Wireless Personal Multimedia Communications*, vol. 2 of *WPMC '02*, pp. 504–508, Oct 2002.

[17] Chambers, B. A., *The grid roofnet: a rooftop ad hoc wireless network*. PhD thesis, Massachusetts Institute of Technology, 2002.

[18] Chang, X., "Network simulations with OPNET," in *Proceedings of the 31st Winter Simulation Conference: Simulation—a Bridge to the Future - Volume 1*, WSC '99, (New York, NY, USA), pp. 307–314, ACM, 1999.

[19] Chen, K. and Nahrstedt, K., "Effective location-guided tree construction algorithms for small group multicast in MANET," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3 of *INFOCOM 2002*, pp. 1180–1189 vol.3, 2002.

[20] Chiang, C.-C., Gerla, M., and Zhang, L., "Adaptive shared tree multicast in mobile wireless networks," in *IEEE Global Telecommunications Conference: The Bridge to Global Integration*, vol. 3 of *GLOBECOM 1998*, pp. 1817–1822 vol.3, 1998.

[21] Chiang, C., Gerla, M., and Zhang, L., "Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks," *Cluster Computing*, vol. 1, no. 2, pp. 187–196, 1998.

[22] Chiu, C.-Y., Wu, E.-K., and Chen, G.-H., "A reliable and efficient MAC layer broadcast (multicast) protocol for mobile ad hoc networks," in *IEEE Global Telecommunications Conference*, vol. 5 of *GLOBECOM '04*, pp. 2802–2807 Vol.5, Nov 2004.

[23] Das, S., Pucha, H., and Hu, Y., "Distributed hashing for scalable multicast in wireless ad hoc networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 347–362, March 2008.

[24] De Couto, D. S. J., Aguayo, D., Chambers, B. A., and Morris, R., "Performance of multihop wireless networks: Shortest path is not enough," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–88, Jan. 2003.

[25] Er, I. and Seah, W., "Distributed Steiner-like multicast path setup for mesh-based multicast routing in ad hoc networks," in *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 2 of *SUTC 2006*, pp. 192–197, June 2006.

[26] Erceg, V., Greenstein, L., Tjandra, S., Parkoff, S., Gupta, A., Kulic, B., Julius, A., and Bianchi, R., "An empirically based path loss model for wireless channels in suburban environments," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1205–1211, Jul 1999.

[27] Erlebach, T. and Grant, T., "Scheduling multicast transmissions under SINR constraints," in *Algorithms for Sensor Systems* (Scheideler, C., ed.), vol. 6451 of *Lecture Notes in Computer Science*, pp. 47–61, Springer Berlin Heidelberg, 2010.

[28] Feng, C.-H., Zhang, Y., Demirkol, I., and Heinzelman, W., "Stateless multicast protocol for ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 240–253, Feb 2012.

[29] Friis, H., "A note on a simple transmission formula," *Proceedings of the IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[30] Garcia-Luna-Aceves, J. and Madruga, E., "The core-assisted mesh protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 1380–1394, Aug 1999.

[31] Giruka, V. and Singhal, M., "Hello protocols for ad-hoc networks: overhead and accuracy tradeoffs," in *Proceedings of the 2005 IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, WoWMoM '05, pp. 354–361, June 2005.

[32] Gómez, D., Agüero, R., García-Arranz, M., and Muñoz, L., "Replication of the bursty behavior of indoor WLAN channels," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, (ICST, Brussels, Belgium, Belgium), pp. 219–226, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.

[33] Gopalsamy, T., Singhal, M., Panda, D., and Sadayappan, P., "A reliable multicast algorithm for mobile ad hoc networks," in *Proceedings of the 22nd International Conference on Distributed Computing Systems*, ICDCS 2002, pp. 563–570, 2002.

[34] Gossain, H., Nandiraju, N., Anand, K., and Agrawal, D., "Supporting MAC layer multicast in ieee 802.11 based MANETs: issues and solutions," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN 2004, pp. 172–179, Nov 2004.

[35] Gui, C. and Mohapatra, P., "Efficient overlay multicast for mobile ad hoc networks," in *IEEE Wireless Communications and Networking Conference*, vol. 2 of *WCNC 2003*, pp. 1118–1123 vol.2, March 2003.

[36] Gupta, P. and Kumar, P., "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, Mar 2000.

[37] Gupta, S. K. S., Shankar, V., and Lalwani, S., "Reliable multicast MAC protocol for wireless LANs," in *IEEE International Conference on Communications*, vol. 1 of *ICC '03*, pp. 93–97, May 2003.

[38] Heidemann, J., Bulusu, N., Elson, J., Intanagonwiwat, C., Lan, K.-c., Xu, Y., Ye, W., Estrin, D., and Govindan, R., "Effects of detail in wireless network simulation," in *Proceedings of the SCS Multiconference on Distributed simulation*, pp. 3–11, 2001.

[39] Ingelrest, F., Mitton, N., and Simplot-Ryl, D., "A turnover based adaptive HELLO protocol for mobile ad hoc and sensor networks," in *15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '07, pp. 9–14, Oct 2007.

[40] Jain, S. and Das, S., "MAC layer multicast in wireless multihop networks," in *First International Conference on Communication System Software and Middleware*, Comsware 2006, pp. 1–10, 2006.

[41] Jetcheva, J. G. and Johnson, D. B., "Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks," in *Proceedings of the 2Nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '01, (New York, NY, USA), pp. 33–44, ACM, 2001.

[42] Ji, L. and Corson, M. S., "A lightweight adaptive multicast algorithm," in *IEEE Global Telecommunications Conference: The Bridge to Global Integration*, vol. 2 of *GLOBECOM 1998*, pp. 1036–1042, 1998.

[43] Ji, L. and Corson, M., "Differential destination multicast-a MANET multicast routing protocol for small groups," in *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2 of *INFOCOM 2001*, pp. 1192–1201 vol.2, 2001.

[44] Jiao, C., Schwiebert, L., and Xu, B., "On modeling the packet error statistics in bursty channels," in *Proceedings of the 27th Annual IEEE International Conference on Local Computer Networks*, LCN 2002, pp. 534–541, Nov 2002.

[45] Junhai, L., Danxia, Y., Liu, X., and Mingyu, F., "A survey of multicast routing protocols for mobile ad-hoc networks," *IEEE Communications Surveys Tutorials*, vol. 11, pp. 78–91, First 2009.

[46] Kernapke, R., Lohs, S., and Nolte, J., "Simulation of unidirectional links in wireless sensor networks," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, SimuTools '14, (ICST, Brussels, Belgium, Belgium), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), March 2014.

[47] Kotz, D., Newport, C., Gray, R. S., Liu, J., Yuan, Y., and Elliott, C., "Experimental evaluation of wireless simulation assumptions," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '04, (New York, NY, USA), pp. 78–82, ACM, 2004.

[48] Koutsonikolas, D., Das, S., Hu, Y., and Stojmenovic, I., "Hierarchical geographic multicast routing for wireless sensor networks," *Wireless Networks*, vol. 16, no. 2, pp. 449–466, 2010.

[49] Kranakis, E., Singh, H., and Urrutia, J., "Compass routing on geometric networks," in *Proceedings of the 11th Canadian Conference on Computational Geometry*, pp. 51–54, 1999.

[50] Lacage, M. and Henderson, T. R., "Yet another network simulator," in *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*, WNS2 '06, (New York, NY, USA), ACM, 2006.

[51] Lee, H., Cerpa, A., and Levis, P., "Improving wireless simulation through noise modeling," in *6th International Symposium on Information Processing in Sensor Networks*, IPSN 2007, pp. 21–30, April 2007.

[52] Lee, S.-J., Su, W., and Gerla, M., "On-demand multicast routing protocol in multihop wireless mobile networks," *Mobile Networks and Applications*, vol. 7, no. 6, pp. 441–453, 2002.

[53] Lertpratchya, D., Blough, D. M., and Riley, G. F., "Interference-aware multicast for wireless multihop networks," in *IEEE Wireless Communications and Networking Conference*, WCNC 2014, April 2014.

[54] Lertpratchya, D., Riley, G. F., and Blough, D. M., "Fast and accurate link discovery integrated with reliable multicast in 802.11," tech. rep., Center for Experimental Research in Computer Systems, Georgia Institute of Technology, 2013.

[55] Lertpratchya, D., Riley, G. F., and Blough, D. M., "Simulating frame-level bursty links in wireless networks," in *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques*, SimuTools '14, (ICST, Brussels, Belgium, Belgium), ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), March 2014.

[56] Li, X.-Y., Tang, S.-J., and Frieder, O., "Multicast capacity for large scale wireless ad hoc networks," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, MobiCom '07, (New York, NY, USA), pp. 266–277, ACM, 2007.

[57] Lui, G., Gallagher, T., Li, B., Dempster, A., and Rizos, C., "Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of wlan localization," in *2011 International Conference on Localization and GNSS*, ICL-GNSS, pp. 53–57, June 2011.

[58] Lun, D., Ratnakar, N., Koetter, R., Medard, M., Ahmed, E., and Lee, H., "Achieving minimum-cost multicast: a decentralized approach based on network coding," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3 of *INFOCOM 2005*, pp. 1607–1617 vol. 3, March 2005.

[59] Maltz, D., Broch, J., and Johnson, D., "Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed," in *IEEE Wireless Communications and Networking Conference*, vol. 3 of *WCNC 2000*, pp. 992–997, 2000.

[60] Maltz, D. A., Broch, J., and Johnson, D. B., "Experiences designing and building a multi-hop wireless ad hoc network testbed," tech. rep., DTIC Document, 1999.

[61] Marina, M. K. and Das, S. R., "Routing performance in the presence of unidirectional links in multihop wireless networks," in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '02, (New York, NY, USA), pp. 12–23, ACM, 2002.

[62] Meghanathan, N., "Determining a sequence of stable multicast Steiner trees in mobile ad hoc networks," in *Proceedings of the 44th Annual Southeast Regional Conference*, ACM-SE 44, (New York, NY, USA), pp. 102–106, ACM, 2006.

[63] Mushkin, M. and Bar-David, I., "Capacity and coding for the Gilbert-Elliott channels," *IEEE Transactions on Information Theory*, vol. 35, pp. 1277–1290, Nov 1989.

[64] Nguyen, U. T., "On multicast routing in wireless mesh networks," *Computer Communications*, vol. 31, no. 7, pp. 1385 – 1399, 2008.

[65] Niculescu, D., "Interference map for 802.11 networks," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, (New York, NY, USA), pp. 339–350, ACM, 2007.

[66] Oh, S. Y., Park, J.-S., and Gerla, M., "E-ODMRP: Enhanced ODMRP with motion adaptive refresh," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1044 – 1053, 2008.

[67] Ozaki, T., Kim, J. B., and Suda, T., "Bandwidth-efficient multicast routing protocol for ad-hoc networks," in *Proceedings of the Eight International Conference on Computer Communications and Networks*, ICCCN 1999, pp. 10–17, 1999.

[68] Park, J.-S., Gerla, M., Lun, D., Yi, Y., and Medard, M., "Codecast: a network-coding-based ad hoc multicast protocol," *IEEE Wireless Communications*, vol. 13, pp. 76–81, October 2006.

[69] Park, S. and Park, D., "Adaptive core multicast routing protocol," *Wireless Networks*, vol. 10, no. 1, pp. 53–60, 2004.

[70] Penttinen, A., "Efficient multicast tree algorithm for ad hoc networks," in *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, MASS 2004, pp. 519–521, Oct 2004.

[71] Perkins, C. and Royer, E., "Ad-hoc on-demand distance vector routing," in *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, WMCSA '99, pp. 90–100, Feb 1999.

[72] Prakash, R., "Unidirectional links prove costly in wireless ad hoc networks," in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, DIALM '99, (New York, NY, USA), pp. 15–22, ACM, 1999.

[73] Ramasubramanian, V., Chandra, R., and Mosse, D., "Providing a bidirectional abstraction for unidirectional ad hoc networks," in *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3 of *INFOCOM 2002*, pp. 1258–1267 vol.3, 2002.

[74] Riley, G. F., "Simulation of large scale networks II: Large-scale network simulations with GTNetS," in *Proceedings of the 35th Winter Simulation Conference: Driving Innovation*, WSC '03, pp. 676–684, Winter Simulation Conference, 2003.

[75] Rodolakis, G., Laouiti, A., Jacquet, P., and Naimi, A. M., "Multicast overlay spanning trees in ad hoc networks: Capacity bounds, protocol design and performance evaluation," *Computer Communications*, vol. 31, no. 7, pp. 1400 – 1412, 2008. Special Issue: Resource Management and routing in Wireless Mesh Networks.

[76] Royer, E. M. and Perkins, C. E., "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, (New York, NY, USA), pp. 207–218, ACM, 1999.

[77] Ruiz, P. and Gomez-Skarmeta, A., "Approximating optimal multicast trees in wireless multihop networks," in *Proceedings of the 10th IEEE Symposium on Computers and Communications*, ISCC 2005, pp. 686–691, June 2005.

[78] Sanchez, J., Ruiz, P., and Stojmenovic, I., "GMR: Geographic multicast routing for wireless sensor networks," in *3rd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, vol. 1 of *SECON '06*, pp. 20–29, Sept 2006.

[79] Shakkottai, S., Liu, X., and Srikant, R., "The multicast capacity of large multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1691–1700, Dec. 2010.

[80] Sheu, S.-T., Tsai, Y., and Chen, T., "A highly reliable broadcast scheme for IEEE 802.11 multi-hop ad hoc networks," in *IEEE International Conference on Communications*, vol. 1 of *ICC 2002*, pp. 610–615, 2002.

[81] Si, W. and Li, C., "RMAC: a reliable multicast mac protocol for wireless ad hoc networks," in *International Conference on Parallel Processing*, ICPP 2004, pp. 494–501 vol.1, Aug 2004.

[82] Sinha, P., Sivakumar, R., and Bharghavan, V., "MCEDAR: multicast core-extraction distributed ad hoc routing," in *IEEE Wireless Communications and Networking Conference*, WCNC 1999, pp. 1313–1317 vol.3, 1999.

[83] Srinivasan, K., Kazandjieva, M. A., Agarwal, S., and Levis, P., "The β-factor: Measuring wireless link burstiness," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, (New York, NY, USA), pp. 29–42, ACM, 2008.

[84] Stoffers, M. and Riley, G., "Comparing the ns-3 propagation models," in *20th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '12, pp. 61–67, Aug 2012.

[85] Tang, K. and Gerla, M., "MAC reliable broadcast in ad hoc networks," in *Military Communications Conference, Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 2 of *MILCOM 2001*, pp. 1008–1013 vol.2, 2001.

[86] te Sun, M., Huang, L., Wang, S., Arora, A., and Lai, T.-H., "Reliable MAC layer multicast in IEEE 802.11 wireless networks," *Wireless Communications and Mobile Computing*, vol. 3, pp. 439–453, 2003.

[87] Toh, C.-K., Guichal, G., and Bunchua, S., "ABAM: on-demand associativity-based multicast routing for ad hoc mobile networks," in *Proceedings of the IEEE 52nd Vehicular Technology Conference*, vol. 3 of *VTC 2000-Fall*, pp. 987–993, 2000.

[88] Vaishampayan, R. and Garcia-Luna-Aceves, J., "Efficient and robust multicast routing in mobile ad hoc networks," in *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, MASS 2004, pp. 304–313, Oct 2004.

[89] Vlavianos, A., Law, L., Broustis, I., Krishnamurthy, S., and Faloutsos, M., "Assessing link quality in IEEE 802.11 wireless networks: Which is the right metric?," in *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, PIMRC 2008, pp. 1–6, 2008.

[90] Wang, K., Chiasserini, C.-F., Rao, R. R., and Proakis, J. G., "A joint solution to scheduling and power control for multicasting in wireless ad hoc networks," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 144–152, Jan. 2005.

[91] Warme, D., Winter, P., and Zachariasen, M., "Exact algorithms for plane Steiner tree problems: A computational study," *Advances in Steiner Trees*, vol. 6, pp. 81–116, 2000.

[92] Wolter, K., Reinecke, P., Krauss, T., Happ, D., and Eitel, F., "PH-distributed fault models for mobile communication," in *Proceedings of the 44th Winter Simulation Conference*, WSC '12, pp. 429:1–429:12, Winter Simulation Conference, 2012.

[93] Wu, C. and Tay, Y. C., "AMRIS: a multicast protocol for ad hoc wireless networks," in *Proceedings of the IEEE Military Communications Conference*, vol. 1 of *MILCOM 1999*, pp. 25–29 vol.1, 1999.

[94] Xiang, X., Wang, X., and Zhou, Z., "An efficient geographic multicast protocol for mobile ad hoc networks," in *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, WOWMOM '06, (Washington, DC, USA), pp. 73–82, IEEE Computer Society, 2006.

[95] Xie, J., Talpade, R. R., Mcauley, A., and Liu, M., "AMRoute: Ad hoc multicast routing protocol," *Mobile Networks and Applications*, vol. 7, pp. 429–439, Dec. 2002.

[96] Yi, J. and Poellabauer, C., "Real-time multicast for wireless multihop networks," *Computers & Electrical Engineering*, vol. 36, no. 2, pp. 313 – 327, 2010. Wireless ad hoc, Sensor and Mesh Networks.

[97] Yi, Y., Gerla, M., and Obraczka, K., "Scalable team multicast in wireless ad hoc networks exploiting coordinated motion," *Ad Hoc Networks*, vol. 2, no. 2, pp. 171 – 184, 2004.

[98] Yoon, J., Liu, M., and Noble, B., "Random waypoint considered harmful," in *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2 of *INFOCOM 2003*, pp. 1312–1321 vol.2, March 2003.

[99] Zeng, X., Bagrodia, R., and Gerla, M., "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation*, PADS 98, pp. 154–161, May 1998.

[100] Zhao, J. and Govindan, R., "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, (New York, NY, USA), pp. 1–13, ACM, 2003.

[101] Zhao, X., Chou, C. T., Guo, J., and Jha, S., "Protecting multicast sessions in wireless mesh networks," in *Proceedings of the 31th Annual IEEE International Conference on Local Computer Networks*, LCN 2006, pp. 467–474, Nov 2006.

[102] Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. A., "Impact of radio irregularity on wireless sensor networks," in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services*, MobiSys '04, (New York, NY, USA), pp. 125–138, ACM, 2004.